GEORGIA INSTITUTE OF TECHNOLOGY — OFFICE OF CONTRACT ADMINISTRATION

PROJECT ADMINISTRATION DATA SHEET

[X] ORIGINAL [ ] REVISION NO. ____

Project No. E-24-635 R6055-0A0 GTRC DATE 10 / 25 / 85

Project Director: J. J. Jarvis School/~~XXX~~ ISyE

Sponsor: Office of Naval Research, 800 North Quincy St., Arlington, VA 22217

Type Agreement: Contract No. N00014-85-C-0797

Award Period: From 9/15/85 To 9/14/86 (Performance) 11/13/86 (Reports)

Sponsor Amount:	This Change	Total to Date
Estimated:	$ 288,324	$ 288,324
Funded:	$ 288,324	$ 288,324

Cost Sharing Amount: $ 15,000 Cost Sharing No: E-24-327 F6055-0A0

Title: Implementation of Advanced Technology for Strategic Mobility Decision Support Systems

ADMINISTRATIVE DATA

OCA Contact R. Dennis Farmer x-4820

1) Sponsor Technical Contact:

Head, Mathematical Sciences Division
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

2) Sponsor Admin/Contractual Matters:

Mr. Thomas A. Bryant
Office of Naval Research
Resident Representative
206 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0490
Phone: 881-4374

Defense Priority Rating: S

Military Security Classification: N/A

(or) Company/Industrial Proprietary: ____

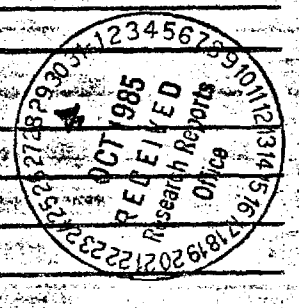
RESTRICTIONS

See Attached Gov't. Supplemental Information Sheet for Additional Requirements.

Travel: Foreign travel must have prior approval — Contact OCA in each case. Domestic travel requires sponsor approval where total will exceed greater of $500 or 125% of approved proposal budget category.

Equipment: Title vests with Georgia Tech will prior ACO approval if less than $5,000. Greater than $5,000 shall vest as determined by the ACO.

COMMENTS:



SPONSOR'S I. D. NO. 02.103.000.86.001

COPIES TO:

Project Director
Research Administrative Network
Research Property Management
Accounting
Procurement/GTRI Supply Services
Research Security Services
Reports Coordinator (OCA)
Research Communications (2)
GTRC
Library
Project File
Other A. Jones

FORM OCA 65-285

GEORGIA INSTITUTE OF TECHNOLOGY OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION/CLOSEOUT SHEET

Date 5-26-87

Project No. E-24-635 School/~~Lab~~ ISyE

Includes Subproject No.(s) N/A

Project Director(s) J.J. Jarvis GTRC / ~~GIT~~

Sponsor Office of Naval Research, 800 North Quincy St., Arlington, VA 22217

Title Implementation of Advanced Technology for Strategic Mobility Decision Support Systems

Effective Completion Date: 9/14/86 (Performance) 11/13/86 (Reports)

Grant/Contract Closeout Actions Remaining:

- [ ] None
- [X] Final Invoice or Final Fiscal Report
- [X] Closing Documents
- [X] Final Report of Inventions – Questionnaire sent to P.I.
- [X] Govt. Property Inventory & Related Certificate
- [ ] Classified Material Certificate
- [ ] Other

ontinues Project No. Continued by Project No.

OPIES TO:

roject Director
esearch Administrative Network
esearch Property Management
ccounting
ocurement/GTRI Supply Services
esearch Security Services
eports Coordinator (OCA)
egal Services
~~Research Communications (2)~~
Library
GTRC
Project File
Other Duane H.
Angela DuBose
Russ Embry

ORM OCA 69.285

Georgia Institute

of

Technology

SEARCH

SCHOOL OF INDUSTRIAL
AND SYSTEMS
ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM
OF GEORGIA
ATLANTA, GEORGIA 30332

Report for:
Joint Depolyment Agency
MacDill Air Force Base, FL 33608

Validation of MODES

John J. Jarvis
H. Donald Ratliff
Principal Investigators

PDRC 86-01

Report by:
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

This work is supported by the Office of Naval Research under Contract No. N00014-85-C-0797 and N00014-83-K-0147. Reproduction in whole or in part is permitted for any purpose of the U. S. Government

1.0 MODES DESIGN REQUIREMENTS.

Central to any validation of MODES are the principal requirements under which it was designed and constructed.

(1) MODES should support crisis action deployment planning.

(2) MODES should aid in the very early part of the crisis action planning process.

(3) MODES should provide useful information in two to four hours.

(4) MODES should address questions concerning "gross transportation feasibility" among alternative courses of action.

A comprehensive understanding of these design criteria is essential in developing a reasonable validation process.

1.1 Supporting Crisis Action Planning

The operative terms here are "support" and "crisis action". MODES was not designed to replace the Supported Commander and his staff. It was intended as a tool for enhancing their capabilities. As a result MODES should be evaluated with planners, not against them.

MODES is not a deliberate planning tool. It was not designed to accomodate the level of detail which is possible in a deliberate planning cycle. It was designed to aid in making rapid go/nogo decisions at the first level in a hierarchical planning system. At its level of use, it should help establish attainable EADs, LADs, and RDDs as well as provide recommendations on channel size, on mode, and on general routing for movement requirements. It should also provides gross feasiblity assessments with regard to closure. MODES was not designed to provide the detail scheduling information currently generated in the deliberate planning process. The output of MODES should be viewed as one of the inputs to the detailed scheduling process.

1.2 Initial Planning Capability

MODES was designed to provide initial planning capability to the Supported Commander and his staff in the very early stages of the crisis action planning process. As a result, MODES must be prepared to work with a variety of levels and quality of data in providing information on various courses of action.

1.3 Rapid Turnaround

One of the strongest design criteria for MODES was that it must provide information on reasonable sized courses of action in two to four hours. This criteria had the single greatest effect on the final design for MODES.

1.4 Gross Measures of Transportation Feasibility

MODES was designed to provide estimates of gross transportation feasibility to the Supported Commander. Once MODES provides gross feasibility estimates, at least two more levels of analyses are required to generate the detailed schedules and corresponding detailed closure estimates required by the TOAs. Models to support these additional levels of analysis are currently being actively developed by the TOAs.

2.0 A STRUCTURE FOR MODES VALIDATION

With the MODES design criteria in mind, we believe that a reasonable MODES validation process should try to answer the following three questions:

(1) Are the MODES recommendations reasonable?

(2) What MODES resolution is practical in reasonable response time?

(3) What is the quality of the MODES recommendations?

Each of these questions and associated analyses will be discussed in turn.

2.1 Reasonability Analysis

Reasonability analysis deals with the ability of MODES to provide results which describe reasonable outputs for given inputs. In a sense, this is an intermediate step between verification analysis and final validation.

2.1.1 Extremal Analysis

In testing a procedure for reasonableness it is generally easier to test the extremes before testing mid-range results. In an extremal analysis of MODES, a number (ten to twenty) of scenarios should be described with the parameters fixed so that the only reasonable MODES outputs are readily predicted by analysts. Examples include small scenarios

with fixed ports, fixed channels, and/or limited assets.

Each resulting MODES output should then be evaluated by knowledgeable planners for reasonableness of response to the constraining input conditions. Each evaluation would be rated "pass" or "fail". Failures would be acompanied by an explanation of the difficulty. Each failure or difficulty would be reviewed by MODES designers for possible explanation or subsequent MODES modification and adjustment.

Extremal analyses test the boundaries of MODES responses to assure valid results in extraordinary situations.

2.1.2 Parametric Analysis

The next step in a reasonability analysis is to evaluate MODES responses to variations in parameters. In this analysis, more involved scenarios should be tested wherein MODES results are not so obviously predictable. The output should be evaluated by a team of planners. The team would attempt to rationalize the MODES output or to point out inadequacies in the logic.

In this analysis the MODES parameters would be varied over a range of values. The objective is to test sensitivity of parameter values to MODES output responses. (This also serves as a "tuning" of the MODES system.) Each report (by the test team) would contain an analysis of the robustness of the MODES model to help define rational responses to varying threats.

2.1.3 Feasibility Analysis

This analysis would help answer the question of "what is reasonable MODES resolution?"

This analysis is accomplished by first applying MODES to a set of test scenarios and then using the MODES output as input to the detailed scheduling process. The detailed scheduling could then either be performed manually or for larger scenarios with the aid of TFE or the TOA scheduling models. The detailed schedules and corresponding detailed closure estimates should then be compared with the output of MODES to determine overall consistency. Note that this is not the same as comparing MODES generated results with simulation generated results. Such a comparison would be meaningless since the models do not utilize the same levels of detail and the simulation models themselves have never been validated. The test described here would test whether given levels of resolution are consistent with simulation models in reporting port/asset loadings and closure estimates.

This testing sequence would be repeated for varying resolution of the MODES model (ports, assets, time). A panel of planners would evaluate MODES output with the resulting simulation results and submit a report comparing and contrasting the results. The idea here is not to charge MODES with the responsibility of tracking simulation results, but to use simulation results as a means for experienced planners to assess relative precision of results. This will provide experienced planners with a framework for making reasonable judgements of desired resolution. It will also indicate whether the input parameters for MODES need to be factored to account for the loss of detail in the data. For example, it may be desirable to have MODES work under the assumption that less than one hundred percent of the capability of a channel is actually usable.

3.0 RESPONSE TIME ANALYSIS

In this series of analyses various sizes of problems are input to the MODES model for solution. By varying numbers of ports, asset types, movement requirements, and aggregation levels, and charting the resulting solution times for the MODES model, an understanding of the response time of the MODES model under varying problem conditions will be obtained.

3.1 Parameter Selection

Not all parameters affect the MODES model in the same manner. Adding asset categories increases the LIFTCAP problem only slightly, while leaving the size of the MRMATE problem unchanged. Adding ports increases the LIFTCAP problem to a greater extent and the MRMATE problem moderately. Adding movement requirements adds only to the MRMATE problem.

The LIFTCAP problem generaly requires much less time to solve than the MRMATE problem. Therefore any parameter change which significantly affect the size of the MRMATE problem will generally result in longer solution times.

3.2 Aggregation Level

The single most significant determinant of problem running time is aggregation level. A slight change in aggregation level of almost any of the variables can cause the MRMATE problem to change considerably in size and produce unacceptable solution times.

3.3 Predicting Problem Size

It is possible to predict problem size given input numbers of ports, asset categories, time periods, and movement requirements. Georgia Tech PDRC Report 84-09 gave a series of formulas to predict LIFTCAP and MRMATE problem size given input data. These formulas should be helpful in judging scenario size for testing responsiveness of the MODES model.

4.0 QUALITY ANALYSIS

Once the limits of the MODES model have been determined and its resolution and robustness evaluated, the final test of its quality and usefulness can be made. This series of tests should evaluate the MODES model in planning situations pitted against currently available planning tools.

4.1 Control Groups

The only reasonable test of MODES is in its ablitiy to support crisis action planning in the first two to four hours. Therefore, tests must be developed to evaluate this capability. One such method would employ control groups. Teams of experienced planners would be divided into two groups and given the same scenerio to evaluate in a two to four hour time limit. One group would be permitted to use MODES while the other (the control group) would not. A number of scenarios should be evaluated with each group having a opportunity to become the control group (thus minimizing group bias). Also, if time permits the groups should be reconfigured to minimize any possibility of individual bias.

4.2 Scenario Planning

The MODES group and the control group should both be presented with identical scenarios simulating a crisis action planning situation. A time limit of two to four hours should be given. Both groups may use any technique available, except that the MODES group must also use MODES and the control group may not use MODES. At the end of the alloted time both groups must provide a recommendation and supporting analyses.

This testing technique should be repeated for several scenarios of varing size and complexity. In some cases even though both groups must hand in their recommendations after the alloted time, they may continue planning for some longer time specified in the analysis. A second set of recommendations would be provided by each group at the end

of the new time limit. This would give an indication of the value of MODES when time is not as critical and would indicate the potential for using MODES in the deliberate planning process.

4.3 Plan Evaluation

For each scenario, boths sets of plans (MODES and control) should be evaluated by a third panel of experienced planners. The panel would assess the relative quality of the two plans. The panel would not be time constrained and could use any technique are analysis available to them to draw their conclusions.

The panel would be required to put the two recomendations on a ordinal scale (indicating which is better) or on a cardinal scale (indicating how much better one is over the other). An example of a simple cardinal scale is "superior, better, equal."

The panel should also provide a written assessment of the two recommendations and supporting analyses.

4.4 MODES Quality Assessment

Panel results would be charted, and these together with the written evaluations provide a basis for a validation of MODES. Some group must evaluate the panel results and results of the earlier studies to write a report on the validation of the MODES model. In the final analysis the resulting judgements will be subjective, but this subjectivity will be minimized by the intermediate reports and panel evaluations.

Georgia Institute

of

Technology

SEARCH

SCHOOL OF INDUSTRIAL
AND SYSTEMS
ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM
OF GEORGIA
ATLANTA, GEORGIA 30332

Report for:
Joint Depolyment Agency
MacDill Air Force Base, FL 33608

Two Decomposition Methods for Intra-CONUS Travel

Ananth V. Iyer
John J. Jarvis
H. Donald Ratliff
Michael A. Trick

PDRC 86-02

Report by:
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

This work is supported by the Office of Naval Research under Contract No. N00014-85-C-0797 and N00014-83-K-0147. Reproduction in whole or in part is permitted for any purpose of the U. S. Government

1. INTRODUCTION

In [1], Jarvis et. al. present a method for deployment planning called System for Closure Optimization and Planning (SCOPE). This method uses Benders' decomposition method to decompose the problem into managable components. This report gives two possible decomposition methods for a related military deployment problem - Intra-CONUS movement.

SCOPE is principally concerned with the movement of movement requirements from ports of embarkation (POEs) to ports of debarkation (PODs), using assets such as airplanes and ships. Given a solution to this problem, it is still necessary to plan the movement of the forces to their respective POEs. This problem is the Intra-CONUS (for CONtinental United States) Travel Problem (ICTP).

One result of SCOPE is an assignment of forces to POEs. For ICTP, it will be assumed that each force is assigned to exactly one POE (if SCOPE splits a movement requirement among multiple POEs then ICTP will have more than one force associated with that movement requirement). Each force is also associated with a unique origin. The movement of the force will be from the origin, through a road or rail network, to the POE. It is assumed that the road and rail networks are essentially uncapacitated so the only constraints are on the throughput capability of the origins and POEs.

2. MATHEMATICAL MODEL

Define $x_{i,t,m}$ to be the tonnage of force i that arrives at its POE at time t employing mode m, where m is either M (for motor) or R (for rail). Let r(i,m,t) be the time that force i arrives at its POE if it leaves its origin at time t using mode m. This function is simply a shift function that models the time needed to move from the origin to the POE.

EXAMPLE: If it takes force i two days to move from its origin to its POE by rail then

$$r(i,R,4) = 6$$
$$r(i,R,5) = 7$$

Let

$O(i)$ be the origin for force i;

$E(i)$ be the POE for force i;

MR_i be the tonnage of force i;

$POE_{j,t}$ be the capability (in tons/day) of POE j at time t;

and $OR_{k,t,m}$ be the capability (in tons/day) of origin k at time t with mode m.

The ICTP can be modelled with the following constraints:

(1) $\Sigma_t \Sigma_m x_{i,t,m} \geq MR_i$ for each force i;

(2) $\Sigma_{i:E(i)=j} \Sigma_m x_{i,t,m} \leq POE_{j,t}$ for each POE j and time t;

(3) $\Sigma_{i:O(i)=k} x_{i,r(i,m,t),m} \leq OR_{k,t,m}$ for each origin k, time t and mode m.

(The notation $\Sigma_{i:E(i)=j}$ means: sum over all forces i with E(i) = j).

Constraint (1) fixes the amount moved of each force to be the force size. Constraint (2) models the throughput capability of each POE during each time period. Constraint (3) models the capacity of each origin during each time period, with separate constraints for

rail and motor movement.

This model can be solved as a linear program. Unfortunately, the model can be quite large. A deployment with 10,000 forces, 50 origins, 50 POEs and a time horizon of 100 days will have 2,000,000 variables and 25,000 constraints. The model is far too large to be solved by a general linear programming code in a reasonable amount of time. The following sections develop two decomposition methods which exploit the embedded structure of the model and solve much smaller problems.

3. TIME / MODE DECOMPOSITION

Define $y_{i,t}$ to be the amount of force i that arrives at its POE at time t. Then $y_{i,t} = x_{i,t,R} + x_{i,t,M}$. The model becomes:

(1') $\Sigma_t\ y_{i,t} \geq MR_i$ for each force i;

(2') $\Sigma_{i\ using\ j} y_{i,t} \leq POE_{j,t}$ for each POE j and time t;

(3') $\Sigma_{i\ from\ k} x_{i,r(i,m),t,m} \leq OR_{k,t,m}$ for each origin k, time t, mode m

(4') $x_{i,t,R} + x_{i,t,M} = y_{i,t}$ for each force i and time t.

This can be solved with Benders' decomposition method. The master problem are constraints 1' and 2'. This problem is a transportation network flow problem. The source nodes of the transportation problem are the forces. The destination nodes are the time expanded POEs. A force is connected to a time expanded POE node if it uses that POE and it can arrive at that time. Hence, the network consists of many smaller networks, one for each POE (see Figure 1).

Given a solution to the master problem, $y_{i,t}'$, the subproblem is:

(3') $\Sigma\ x_{i,r(i,m),t,m} \leq OR_{k,t,m}$;

(4') $x_{i,t,R} + x_{i,t,M} = y_{i,t}'$.

This is also a transportation network flow problem. The source nodes are the time expanded movement requirements and the destination nodes are the time expanded origin nodes, one set for each mode. A time expanded force is connected to exactly two origin nodes: one for the corresponding time expanded rail origin and one for the corresponding time expanded motor origin. In other words, if a force node corresponds to arriving at its POE on day 5 and rail travel is two days and motor travel is one day, then the day 5 force node will be connected to the day 3 rail node and the day 4 motor node (see Figure 2). This network also consists of many smaller networks, one

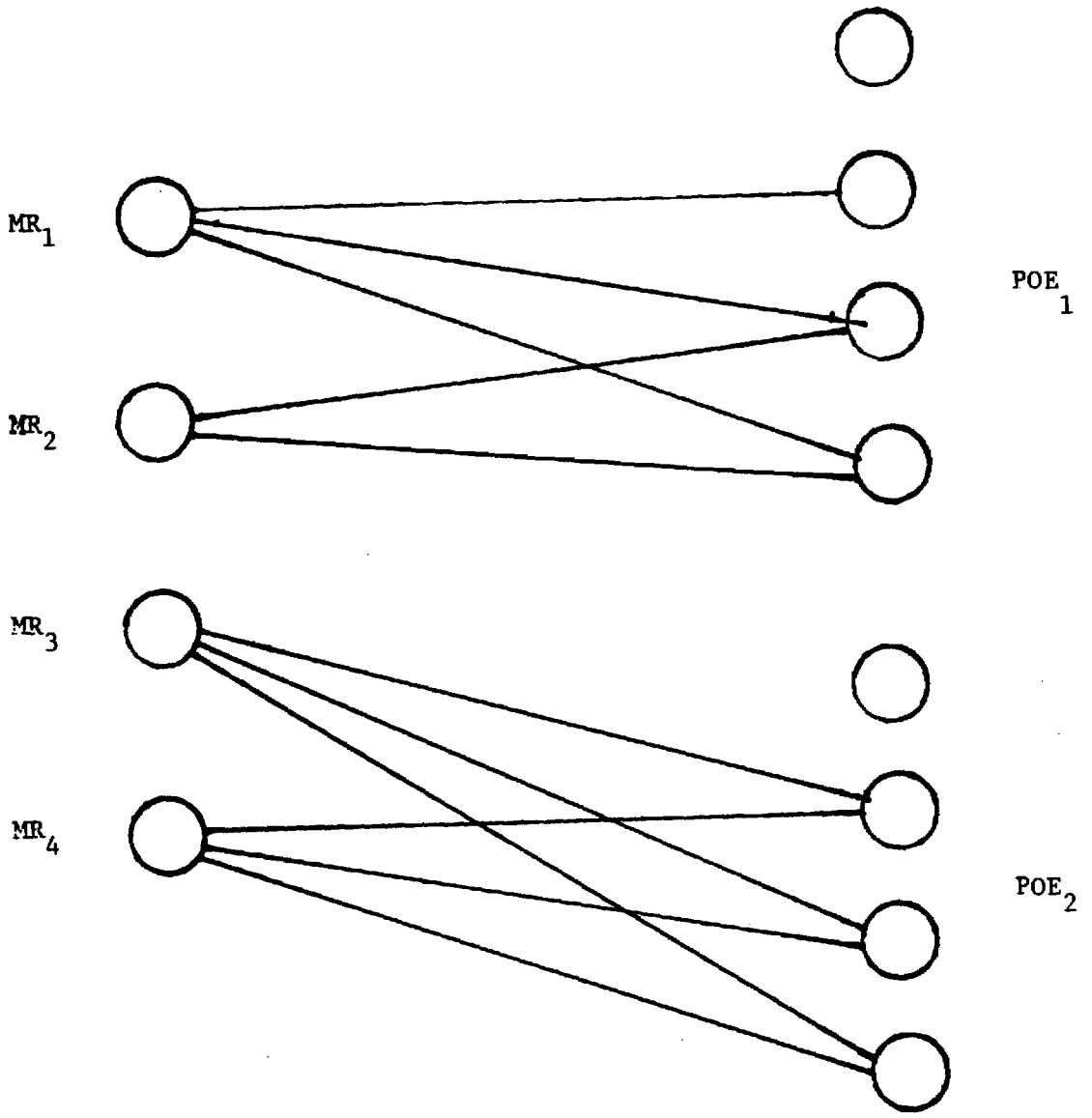


Figure 1. Master Problem for TIME / MODE Decomposition

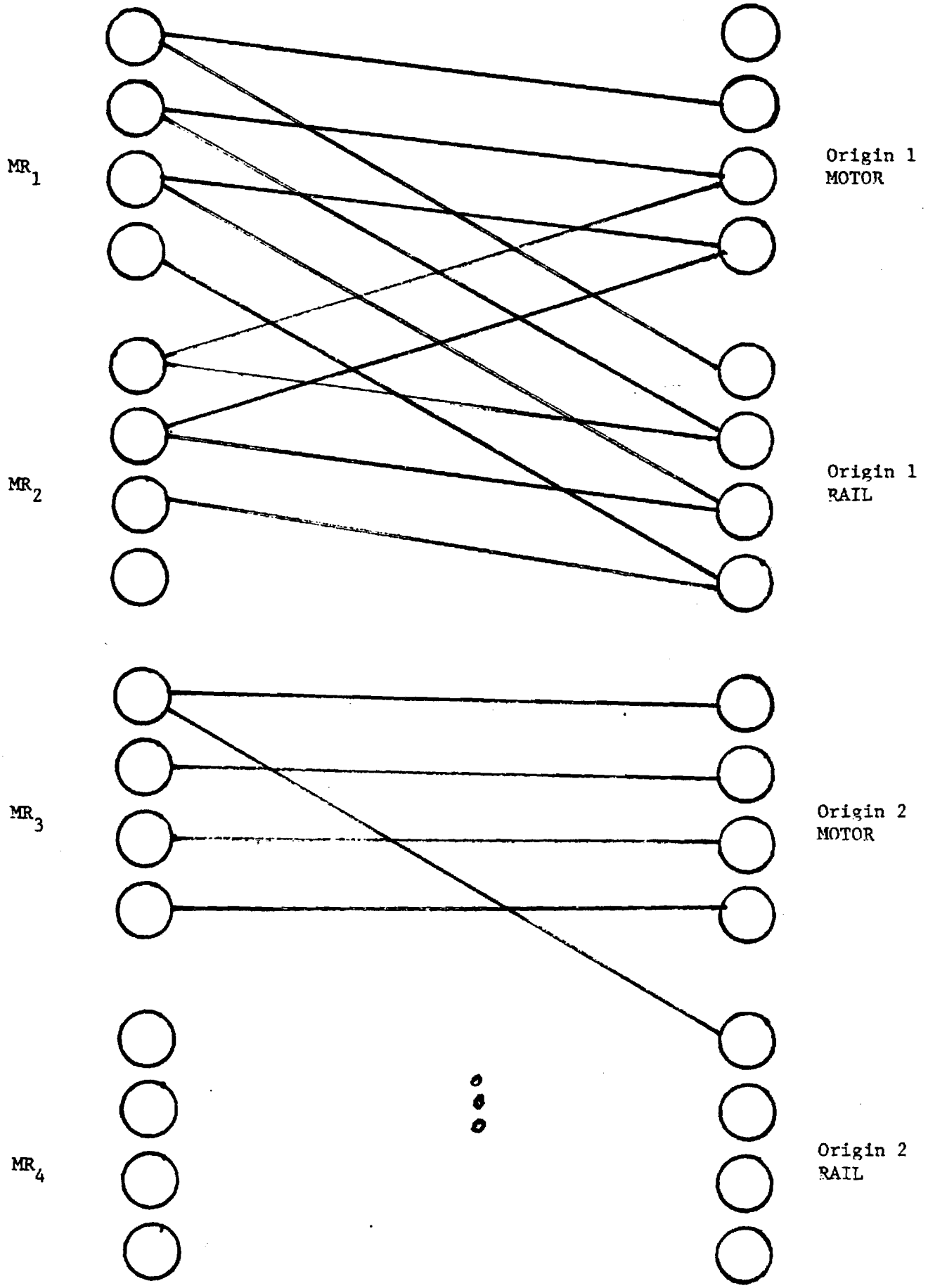


Figure 2. Subproblem for TIME / MODE Decomposition

for each origin. Note that many of the force nodes will have supply zero (for the supply corresponds to $y_{i,t}$') further simplifying the network.

The solution to the subproblem generates a new constraint which is will be added to the master problem. Iteration between the problems continues until a "good" answer is generated.

A simple interpretation is available for this decomposition. The master problem decides how many tons of each force will arrive at its POE on each day, ignoring the origin constraints. The subproblem uses this result and decides whether to use rail or motor movement to avoid congestion at the origin. This decomposition will be called the TIME / MODE decomposition.

4. CHANNEL/ASSIGNMENT DECOMPOSITION

An alternative decomposition strategy is to create channels between origins and POEs and then to allocate forces to the channels. This decomposition assumes that the transportation time is dependent on only the mode, POE and origin and is independent of the force involved.

Define $y_{k,j,t,m}$ to be the tonnage moved from origin k to POE j at time t via mode m. Let q(k, j, m, t) be the time a force will leave origin k if it arrives at POE j at time t via mode m. This gives the following model:

(1") $\Sigma_t \Sigma_m x_{i,t,m} \geq MR_i$ for each force i;

(2") $\Sigma_{\{i : g(i)=j \text{ and } o(i)=k\}} x_{i,t,m} = y_{k,j,t,m}$ for each POE j, origin k, time t and mode m;

(3") $\Sigma_k \Sigma_m y_{k,j,t,m} \leq POE_{j,t}$ for each POE j and time t;

(4") $\Sigma_j y_{k,j,q(k,j,m,t),m} \leq OR_{k,t,m}$ for each origin k, time t and mode m.

Again, this model can be solved with Benders' decomposition method. The master problem consists of constraints 3" and 4". Constraint 3" forces the amount arriving at each POE during each time period to be less than the capacity of the POE. Constraint 4" limits the amount leaving an origin to the origins capacity.

This problem is a transportation network flow problem; the origins are the sources and the POEs are the sinks. Unlike the TIME / MODE decomposition this problem does not break into smaller problems. (see Figure 3).

The subproblem consists of constraints 1" and 2". Given a solution to the master problem ($y_{k,j,t,m}$), the subproblem attempts to allocate the forces to the created channels (origin - POE pairs). Again, this subproblem is a transportation network flow problem; the

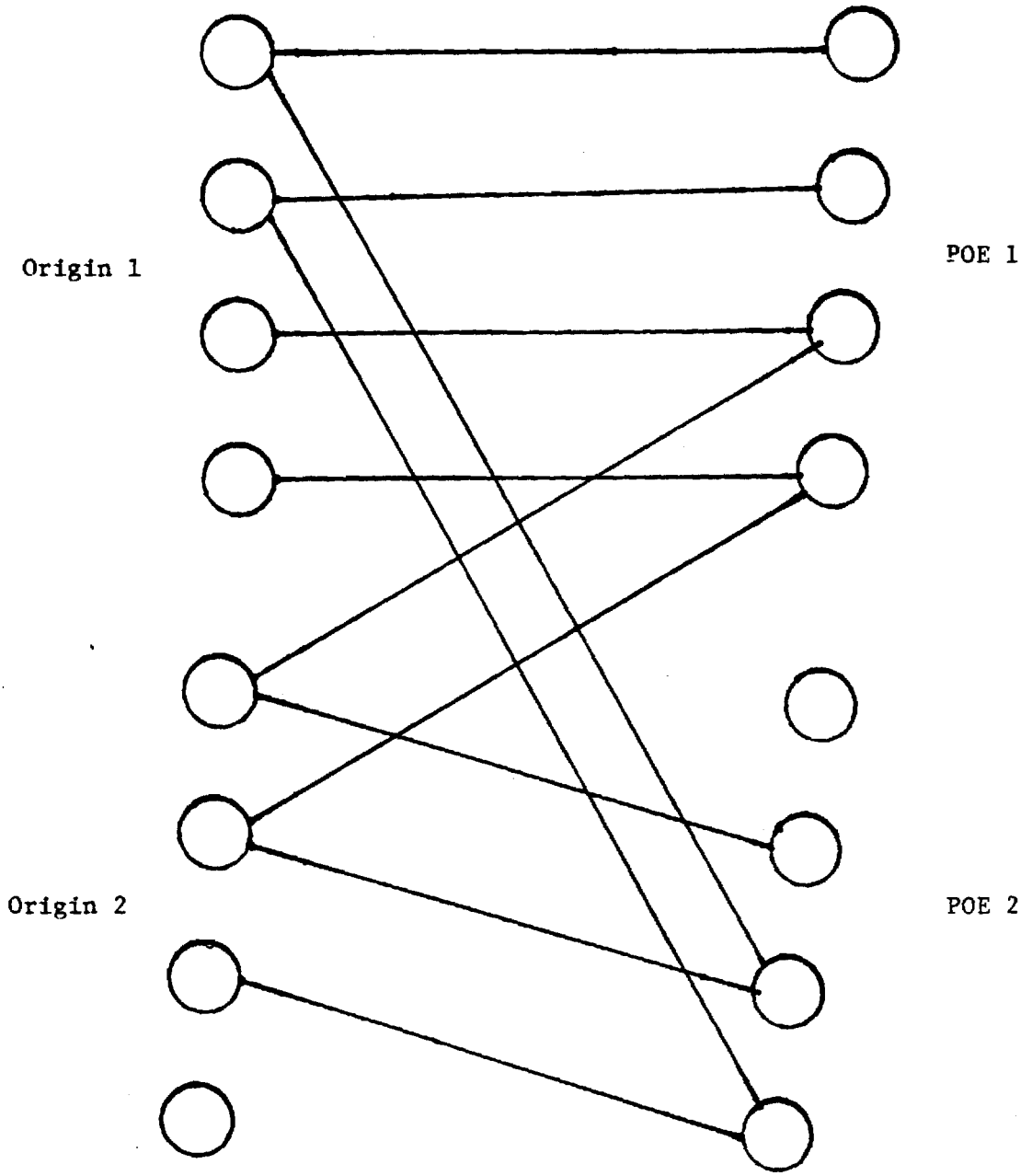


Figure 3. Master Problem for CHANNEL / ASSIGNMENT Decomposition

forces are sources and the channels are sinks. This problem consists of numerous subnetworks. The source nodes in each subproblem consist of those forces with a common origin and POE. The sink nodes correspond to channels opened in the master problem. In general, the subnetworks will be very small (see Figure 4).

A solution to the subproblem will create a constraint in the master problem and the process iterates. This decomposition will be called the CHANNEL / ASSIGNMENT decomposition since it creates channels and then assigns forces to them.

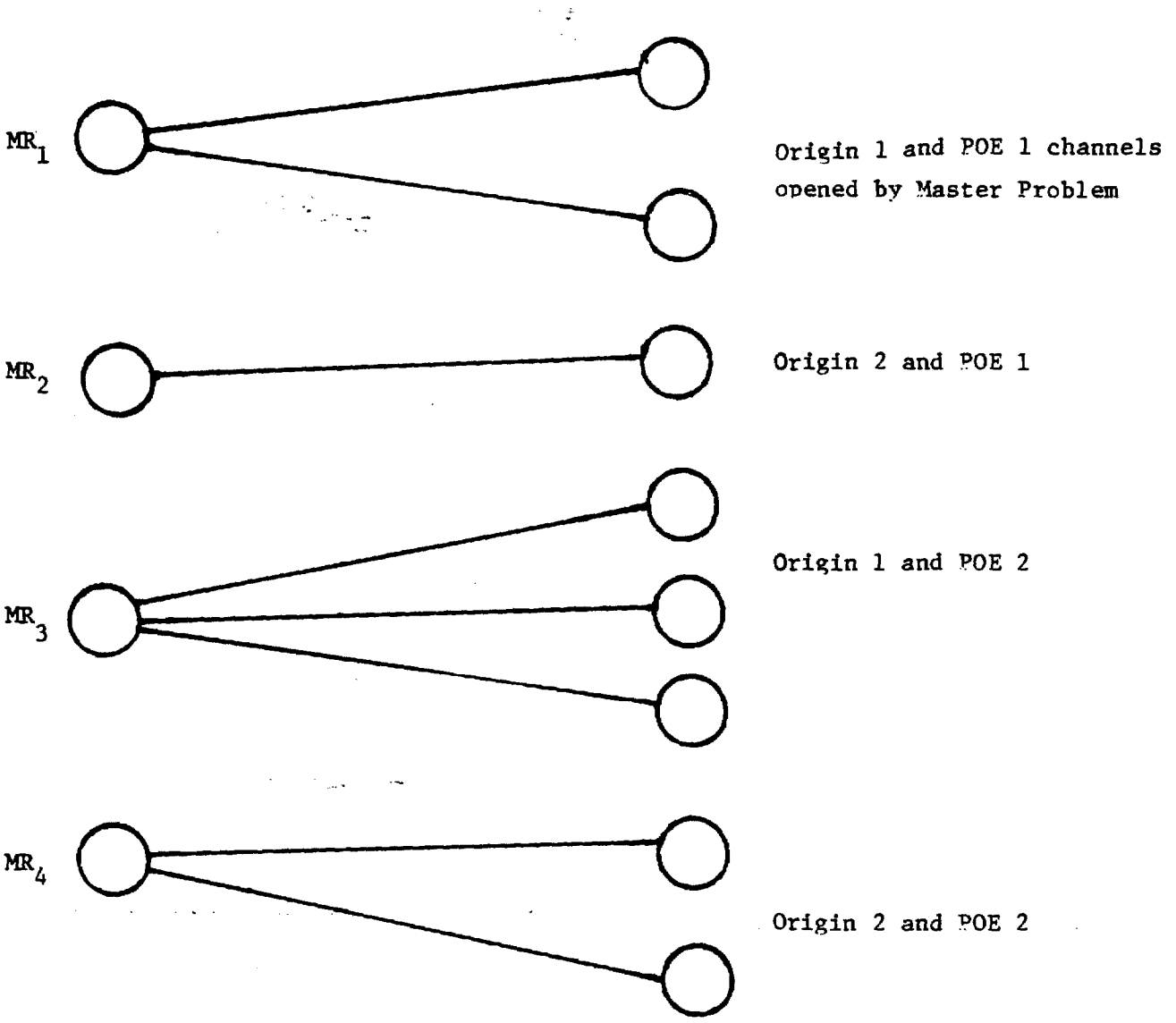


Figure 4. Subproblem for CHANNEL / ASSIGNMENT Decomposition

5. COMPARISON BETWEEN DECOMPOSITIONS

Both decompositions separate ICTP into two or more transportation network problems, which are very efficiently solvable. The sizes of the problems are very different. Consider a problem with 10,000 forces, 50 POEs, 50 origins and a time horizon of 100 days.

The master problem of the TIME / MODE decomposition will have 15,000 nodes and 1,000,000 arcs. Although this problem consists of 50 subnetworks of roughly equal size, the constraints generated by Benders' decomposition method will link these networks, preventing individual solution. The master problem of the CHANNEL / ASSIGNMENT decomposition has 15,000 nodes. The number of arcs is dependent on further details of the problem (how many origin-POE pairs have at least one force), but is certainly no more than 500,000 and may be much less.

The subproblem of the TIME / MODE decomposition has 50 networks with roughly 500 nodes and 1,000 arcs each. The subproblem for the CHANNEL / ASSIGNMENT decomposition consists of many very small networks. The size and number of these networks is dependent on problem data, but will probably be hundreds or thousands of networks with roughly ten nodes and fifty arcs.

Both master problems are very large, and the presence of the Benders' constraints requires the use of a network flow with side constraints method. This method is slower than a standard network flow method. Fortunately, the number of side constraints will be very small (one per iteration) so these problems are still solvable. Further assumptions on the problem (like hard windows for the forces) will further reduce the problem sizes.

Without extnesive testing, it is not possible to determine which

of the two decompositions will provide the better answer for a small number of iterations. Both methods seem to create reasonable sized subproblems, so both are possibilities for further testing.

6. USING THE MODES SOLUTION

This section discusses various ways in which MODES solution information can be used as input to the Intra-Conus models presented in this report. The POE assignment, generated by MODES, is required information which MODES would have to provide to either Intra-Conus model. The remainder of the information discussed in this section is desired information, generated by the MODES model, which could be used in a mathematical programming based decomposition model or in other model types (such as a deterministic simulation approach).

The solution generated by MODES allocates forces to specific time expanded channels. The Intra-Conus models discussed in this report establish movement of forces to the POEs corresponding to these channels. Hence the single piece of information that the Intra-CONUS models, described in this report, definitely need is allocation of forces to POEs.

Depending on the type of Intra-Conus model used, various other pieces of information generated by MODES could be used to accelerate the solution process. In a mathematical programming based decomposition approach similar to those discussed in earlier sections of this report, certain additional information would help in the description of the objective function or as an advanced start procedure.

Dual variables for the time expanded channels would provide costs associated with co-ordinating the Intra-Conus solutions with the MODES solutions. The duals for the channels used would be zero while those for channels not open would be high. This cost information could be used along with the transport cost information as the objective of the decomposition models. Appyling the dual variable information to the Intra-CONUS objective would have the effect of

making it beneficial for the Intra-CONUS models to gravitate towards a solution similar to MODES. In fact, ignoring any differences in costs of rail and motor as well as outload limitations, applying these dual values in the Intra-CONUS objective of either decomposition model would result in the MODES POE allocation being an optimal one.

The effective time windows at the POEs for each force, based on the time windows supplied by the Supporting Commander, could be used in the objective function of the decompositions models by using costs of zero for arriving within these windows and increasing costs for arriving before or after these window areas.

An initial solution regarding time of arrival of a force at its destined POE could be provided by MODES. This time-of-arrival information could be used as a starting solution, to be improved upon in subsequent iterations of the decomposition models.

Useful information can be provided by MODES to other types of Intra-Conus models (e.g., deterministic simulations) as follows.

The time a force arrives at its POE in the MODES solution could be used as a target solution to be aimed for by the model.

MODES might provide information regarding alternative POE choices for a force. This could be accomplished by providing alternative optimal force POE allocation scenarios, or by providing information regarding those force channel allocations that are "close" to each other.

Other types of information would be POE "equivalence" relations. This "equivalence" could be established in a varitey of ways. One way would be based on the forces that MODES allocates to the POE over time. Another way might be by averaging the dual variables for the time expanded channels associated with the POE. This provides

additional flexibility to Intra-Conus models by allowing (limited) choice of a POE assignment which result in the least transport cost.

REFERENCES

[1] Jarvis, J.J., H.D. Ratliff, D.E. Eisenstein, A.V. Iyer, W.G. Nulty, and M.A. Trick, "SCOPE: System for Closure Optimization and Planning Evaluation", PDRC Report 84-09, Georgia Institute of Technology, 1985.

Report for:
Joint Deployment Agency
MacDill Air Force Base, FL 33608

Generalized Network Implementations

John J. Jarvis
H. Donald Ratliff
Michael A. Trick

PDRC 86-03

Report by:
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

This work is supported by the Office of Naval Research under Contract No. N00014-85-C-0797 and N00014-83-K-0147. Reproduction in whole or in part is permitted for any purpose of the U. S. Government

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Generalized networks are an important class of optimization models, with uses in a wide variety of fields. This report describes the development and implementation of a generalized network algorithm.

In [3], Jarvis et. al. recommend a generalized network model, system for closure optimization and planning (SCOPE), for crisis action deployment planning. In SCOPE, large generalized networks must be repeatedly solved. These networks have special structure, which results in computational advantages.

In this report, a generalized network implementation is developed for solving very large generalized networks. This implementation includes new data structures for storing the basis, in-core/out-of-core handling of the arcs, and special handling of pure network structure.

In this report, a detailed examination of the SCOPE model is provided and its effect on implementation issues is discussed. The SCOPE model is highly structured. This report demonstrates how this structure can be used to advantage. In a companion report [4], extensive testing is presented which addresses the question "What affects the computation time for a SCOPE model?"

Section 2 provides an introduction to modeling with pure and generalized networks. Section 3 gives an overview of linear programming, as it applies to generalized network solution methods. In section 4, the special structure of a generalized network basis is detailed and efficient storage methods are developed. Section 5 gives methods for handling the arc data. This includes both storage

methods and techniques for determining the arc to enter the basis. A method for determining the arc to leave the basis is given in section 6. Section 7 describes, in detail, the algorithms used to update the basis structures. Section 8 discusses some related implementation concerns. These include the initial basis to be used, and the effect of embedded pure network structure. Section 9 gives a review of the conclusions.

2.0 MODELING WITH GENERALIZED NETWORKS

Generalized networks, as the name suggests, are a generalization of standard, or "pure", networks. By modifying a restriction that occurs in pure networks, many previously intractable problems have been modeled. Section 2.1 provides a brief overview of pure network modeling. Section 2.2 expands this overview to include generalized networks. Section 2.3 details the SCOPE generalized network model. The SCOPE model is an example of generalized network model. It will form the basis for the examples used later in this report.

2.1 Pure Networks

A "pure" network can be thought of as a pipeline system. This system has suppliers and users of the materiel flowing through the pipe. Each of the suppliers and users has a known supply and demand respectively. The pipeline connects the suppliers and users. The pipeline may have intermediate junction points which are not suppliers or users. (An example would be a pumping station.)

Each pipe has a known capacity. This capacity represents the limit on the pipe expressed in the rate of flow of materiel through the pipe. There is a unit cost associated with materiel that flows through the pipe. This cost is linear in the quantity of materiel; in other words, if the amount flowing through a pipe is doubled, then the cost is also doubled.

The objective is to move the materiel from the suppliers to the

users through the pipeline at minimum cost. This involves assigning flows to pipes so that (1) each user gets the amount needed; (2) no supplier sends more than available; (3) no pipe has more materiel than its capacity; (4) no materiel enters or leaves the system except at users or suppliers; and (5) what enters the pipe at one end, leaves it at the other.

The suppliers, users, and junctions where two or more pipes come together may be represented by points, called nodes. The lengths of pipes between points (nodes) are called arcs. Associated with each node is a number, called its requirement. If the requirement is negative, then the node is a supplier, and the number is the amount that it can supply. If the requirement is positive, then the node is a user, and the number represents the amount it demands. A zero requirement can be used for junctions where arcs meet without representing a supplier or user.

Associated with each node is a flag. The flag indicates whether the requirement must be met exactly, or whether the absolute value of the requirement represents an upper bound for the supplier (or user). This accommodates models with nodes in which suppliers must ship the full amount of their supply and for users that may or may not use the full demand indicated for them.

Arcs have capacities, cost and direction. The node that an arc begins at is called its "TAIL". The ending node is the "HEAD". For arc number ARC these two ends are referred to as TAIL(ARC) and HEAD(ARC) respectively.

Figure 2-1 shows a sample network. The circles represent nodes. The lines (arrows) between them are the arcs.

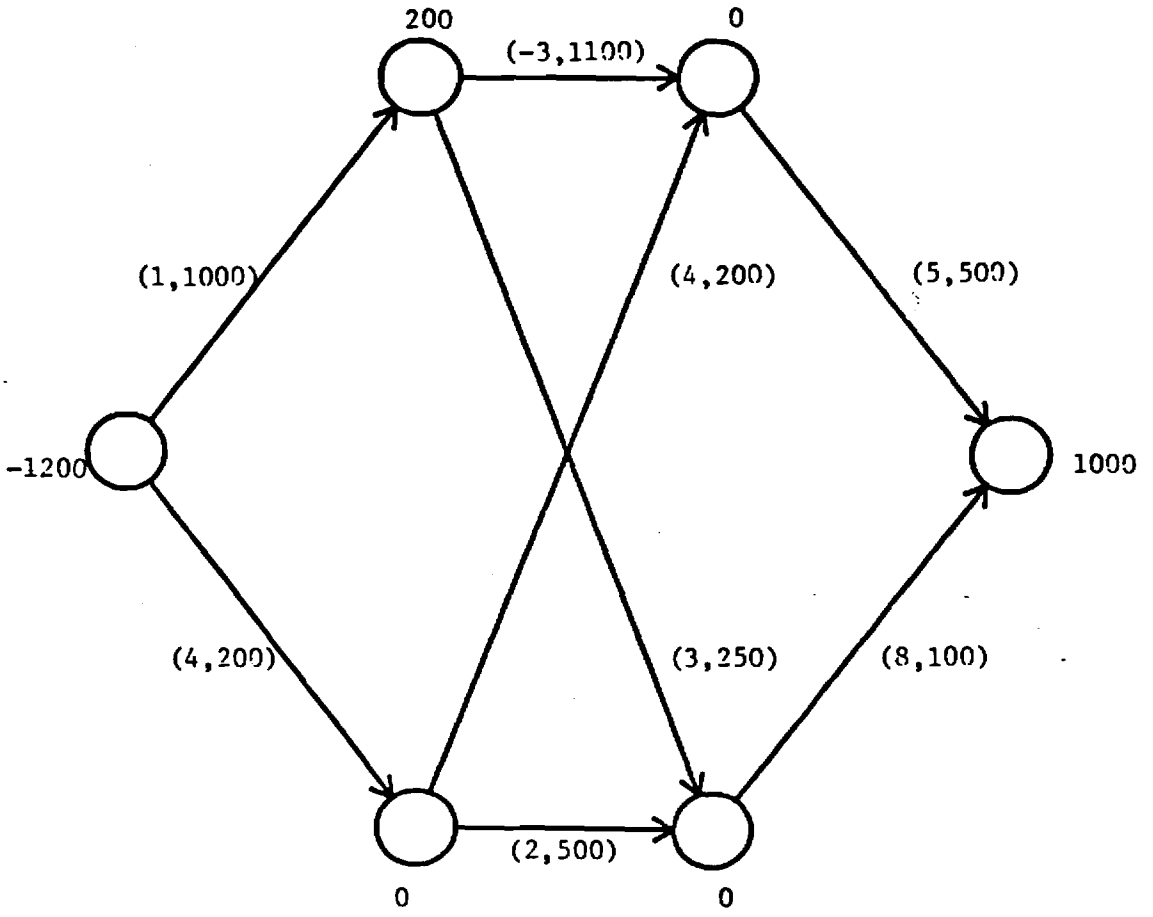


Figure 2-1. Network Representation

2.2 Generalized Networks

Generalized networks are extensions of pure networks. The difference is that the restriction that "flow into an arc equals flow out of the arc" is relaxed. Instead, a generalized network allows for "leaky" arcs or arcs that gain flow. The loss or gain is specific to each arc and can vary throughout the network. The only requirement is that the arc must gain or lose a constant fraction of its flow. For instance, an arc might always triple its flow; another pipe might always quarter its flow. The fraction that the arc changes the flow is called the multiplier. The multipliers for the example arcs are 3 and 0.25 respectively. As a matter of convention, costs are calculated by multiplying the flow that enters the arc by the arc cost.

Because of gains and losses in a generalized network, it is impossible to require that supply equals demand, as is normal in pure network models. "Slack" and "surplus" arcs are required to model the excess supply or demand that normally occurs in a generalized networks. These "slack" and "surplus" arcs are modeled using "self-loops" at each node. "Self-loops" are arcs that begin and end at the same node. Surplus arcs, associated with demand nodes, have multipliers of +1. Slack arcs, attached to supply nodes have multipliers of -1.

All other properties of pure networks hold for generalized networks. A generalized network with all multipliers equal to one is a pure network. An arc with a multiplier of one is called a pure arc.

Generalized network models are very useful in many problems.

1) In a financial model, materiel flowing in a network represents money and nodes represent various points in time. Multipliers can be used to represent increase in money due to interest.

2) In an energy allocation model, materiel flowing in a network represents electricity, and arcs represent physical wires. Multipliers can be used to model energy losses that result when electricity flows along a wire.

3) In a deployment model, materiel flowing in the network are men and equipment to be moved. Arcs represent movement, by either airplane or ship. If materiel is moved by air then the weight (STONS) of the materiel determines the amount that can be moved. If movement is by sea, then the volume (MTONS) or square footage of the movement requirement is critical. Multipliers can be used to model conversion of weight into volume.

2.3 The SCOPE MRMATE Model

In [3], a method for solving a large deployment problem was presented. This method, called System for Closure Planning and Evaluation (SCOPE), addressed the following problem:

Given a set of assets (airplanes and ships), a set of movement requirements (people, ammunition, etc.), and a set of ports to use, is there a way to move the movement requirements with the available assets through ports so that the requirements arrive at the target area when needed?

A much more detailed examination of the problem is provided in [3]. The method proposed is based on three optimization components: a network flow with side constraints which assigns assets to pairs of

ports; a generalized network flow which assigns the movement requirements to assets; and a Benders' constraint generator to link together the two models. This report is concerned only with the second of the optimization pieces.

Fundamental to the SCOPE method is the concept of channels. A channel consists of a pair of ports, together with a number of identical assets. One port, the Port of Embarkation (POE), is where the movement requirements will be loaded onto the assets. The other port, the Port of Debarkation (POD), is the destination of the assets. The assets are assumed to cycle between the POE and POD. Depending on the distance between the POE and POD, the assets may be able to make one or more trips between the POE and POD in a single time period (which can be taken as a day for simplicity). Or, if the ports are far apart or the asset moves slowly, it may take several days to cycle between the POE and the POD. The capability of the channel is the rate at which the assets deliver movement requirements to the POD from the POE.

The first optimization model in SCOPE determines the channel capabilities. The second, generalized network model, must assign the movement requirements to channels at specific time periods so as to meet strategic objectives.

Suppliers in the SCOPE model are the movement requirements. Users are time expanded channels. Time expansion refers to the fact that a channel can move a given amount on each day. Nodes will be created for the channel for each day.

Certain movement requirements cannot be moved on specific channels. For instance, a movement requirement may not be air transportable, so they cannot use channels using air assets. There

are many other restrictions. There is a certain delay involved in getting a movement requirement to the POE, so even if a movement requirement can use a channel, it may not be able to use it on certain days. Arcs are created from each movement requirement to each time expanded channel node that the movement requirement can use. (See Figure 2-2).

The interpretation of flow on an arc from movement requirement M to channel C on day D is that amount of M will arrive at the POE associated with C on day D and will use the assets associated with C to be transported to the POD of C. From this information, it is possible to determine when M will arrive at its final destination. The cost associated with an arc depends on the value of getting M to its final destination at that time.

There are many ways of assigning costs to the arcs, but one of the most flexible involves time windows. Each movement requirement has a window of days in which it is desired to arrive at its final destination. If it arrives at its final destination within its window there is no cost. If it arrives outside its window (either early or late) then the cost is a function of how many days early or late it arrives.

The final complication is in how to measure the size of a flow. There are two types of channels: air and sea. The major limitation on the amount an air channel can move is weight (STONS) of the movement requirements. The major limitation on the sea channels is volume (MTONS). Each movement requirement has a weight and volume, and the relationship between these two values depends on the movement requirement.

To model this, the weight of the movement requirement is taken as

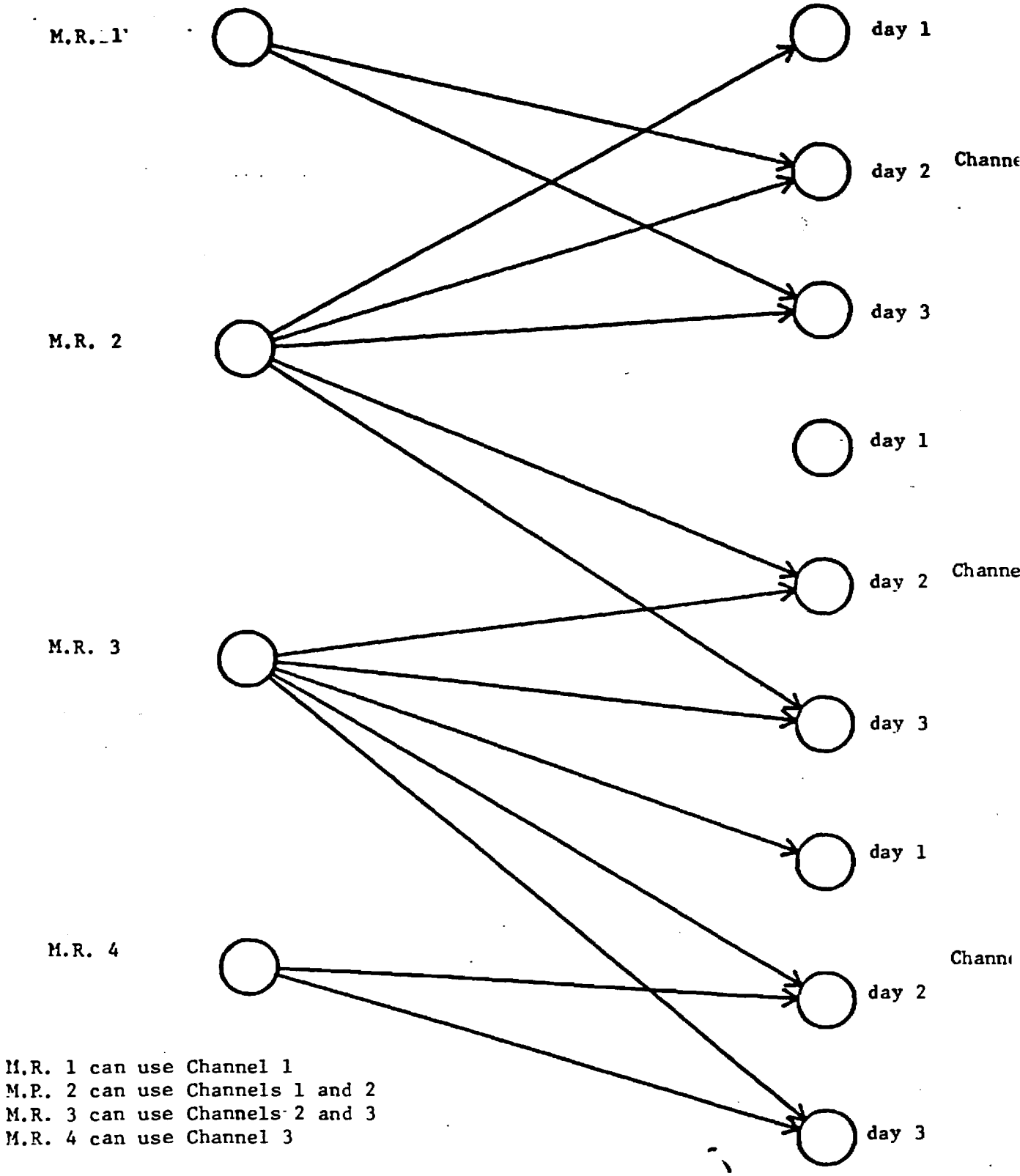


Figure 2-2. SCOPE Basic Model

its size. If the movement requirement is sent by an air channel, no conversion takes place. If the movement requirement is sent by a sea channel, the weight is converted to volume by a multiplier on the arc. For instance, if a movement requirement weighs 20 STONS and has volume 50 MTONS, arcs to air channels will be pure arcs (have multiplier one) and arcs to sea channels will have multipliers of 2.5 (=50/20). Figure 2-3 provides the complete network for the example.

The network created, called the MRMATE network, has significant structure. The major features are:

1) all nodes are either suppliers or users. In fact, this network is a transportation network (see Bazaraa and Jarvis [1]).

2) many of the multipliers are one.

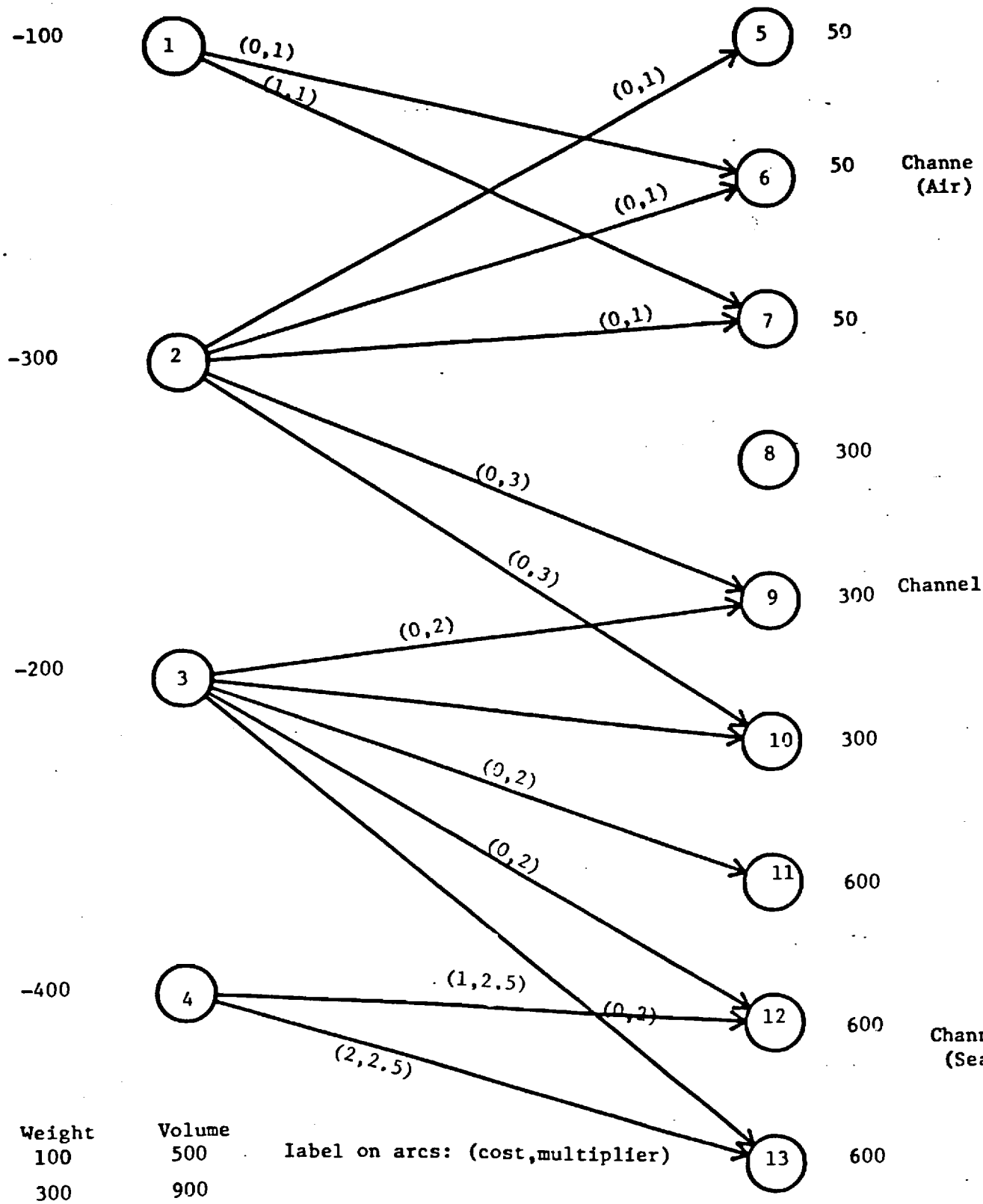
3) all arcs out of a movement requirement have one of two multipliers.

3) many arcs have zero cost.

4) the arcs have no capacities.

The largest problem that could occur in practice is estimated to have approximately 2000 nodes and over 500,000 arcs. Furthermore, these problems must be solved repeatedly in a very short amount of time.

The MRMATE model will be used as an example of the type of specializations possible in implementations of generalized networks in the remainder of this report.



	Weight	Volume
MR1:	100	500
MR2:	300	900
MR3:	200	400
MR4:	400	1000

Iabel on arcs: (cost,multiplier)

Figure 2.3. SCOPE Generalized Metwork Model

3.0 SOLVING GENERALIZED NETWORKS

Since the generalized network model is so useful, it is important to have a computer solution technique that will find solutions quickly. It can be shown that generalized networks are a special case of linear programming, so that any technique to solve linear programming, like the Simplex method, can be used to find solutions to this model. But there are disadvantages of using these general purpose algorithms. The best known methods take too much space and are relatively slow. Fortunately, the Simplex method can be specialized so as to take advantage of the special structure in a generalized network. The specialized simplex method solves generalized networks quickly using little space.

Section 3.1 Linear Programming

Linear programming is the most fundamental optimization model in operations research. Bazaraa and Jarvis [1] provide an excellent introduction to this field. The linear programming model employs the optimization of a linear function subject to a set of linear constraints. A linear function is a function that is of the form:

$C_1 x_1 + C_2 x_2 + ... + C_n x_n$

where each of C_1, C_2, ... C_n are constants

and x_1, x_2, and x_n are the variables.

A linear constraint is of the form:

$A_1 x_1 + A_2 x_2 + ... + A_n x_n = B$

where A_1, A_2, ... A_n and B are all constants.

When a generalized network is represented as a linear program,

variable x_i is associated with each arc. This variable represents the amount of flow in the arc.

There is a linear constraint for each node. This constraint controls the amount of the flow that exits or enters the node. The B value for the constraint is the supply or demand for that node. The constraint forces the net amount of flow at a node (including the self-loop), to be the supply or demand for the node.

Every arc is associated with just two nodes: the tail and head nodes for the arc. This implies that the variable associated with each arc occurs in at most two constraints in the linear program. Self-loops occur in only one constraint.

In matrix terms, each constraint represents a row of the constraint matrix and each variable represents a column. The preceding argument indicates that there are at most two non-zero elements in each column of the constraint matrix.

Because the definition of generalized networks in this report allows just a single multiplier for each arc, one of the non-zeros of each column will be the multiplier on the arc. For arcs that are not self-loops, the other non-zero element of the column will be -1. The multiplier will be in the row associated with the constraint on the head of the arc. The -1, for non-self-loops, is associated with the tail of the arc.

Every variable is assumed to be constrained to be nonnegative. There are standard "tricks" to transform variables not of this form to the assumed form.

Since the arcs (variables) have capacities, it is necessary to treat this problem as a linear program with upper bounds. These upper bounds are linear constraints themselves. Due to the simplicity

of upper bounds it is possible to treat them implicitly in the solution algorithm.

3.2 Solving Linear Programs - The Primal Simplex Method

There are many methods for solving linear programs. The most widely used is the primal simplex method. This technique has proved to be efficient, both in execution time and computer space.

For every linear program, there is an optimal solution with no more than one non-zero variable for each constraint. This is referred to as a basic optimal solution. The optimal non-zero variables form a basis. A basis is any set of variables with the following properties:

1) There are not more variables than constraints in the linear program.

2) No column of the constraint matrix for any variable in the basis can be written as a weighted combination of the columns of the other variables in the basis.

3) There is a feasible solution to the linear program using just the variables in the basis.

The steps of the primal simplex method are as follows:

1) Find an initial basis.

2) Find a variable, not in the basis, to enter the basis. If none exists, STOP. The current basis is optimal.

3) Find the variable in the basis that will leave.

4) Update the basis

5) GOTO Step 2.

One iteration of steps 2 through 4 is called a "pivot".

Step 1 can be accomplished in various ways. The simplest method is to take the "slack" and "surplus" variables that often occur in a linear program and use them as the initial basis. Sometimes artificial variables must be added where "real" slack and surplus variables do not exist. These artificial variables are given a high cost, so that the optimal solution will not employ any of them.

It is often possible to determine a set of variables that creates a very good solution. This usually reduces the number of pivots required to reach optimality. The time to find a good starting solution, called an advanced start, must be short enough not to offset the reduced computation time for the rest of the algorithm.

Identifying a variable to enter the basis is accomplished by determining the change in objective function if the variable is increased by a small value. This change is called the reduced cost.

Increasing the value of the variable entering the basis will change the values of the current basic variables. One of these variables will be the first to reach zero. This is the variable to exit the basis.

The new basis consists of the old basis, without the exiting variable, and the entering variable. Various values must be updated, including the new variable values and the reduced costs for variables not in the basis.

The primal simplex method can be adapted for upper bounds on variables. Rather than treat the upper bounds as "normal" constraints, which would be inefficient, the definition of basis is slightly redefined. A non-basic variable can now have value of either zero or its upper bound. A basic variable can have any value between zero and its upper bound. A non-basic variable at its upper

bound may enter the basis if decreasing its flow slightly improves the objective function. When the basic variables change value (Step 3) one of them will reach its upper bound or zero first. That variable will be the exiting variable.

4.0 STORING THE BASIS

The main reason that a specialized simplex method is faster than a general purpose simplex method for generalized networks is that the basis has a special structure. This structure makes every simplex computation easier. This section defines the basis structure and gives data structures to efficiently store it.

4.1 Basis Definition

A basis in linear programming consists of a set of columns, one for each row, with the property that no column is a weighted sum of the others. In a generalized network, columns correspond to arcs, so the basis is a set of arcs. Rows correspond to nodes, so there is one arc in the basis for each node. The final property, called linear independence, is more complicated to describe.

If a set of arcs is examined, the set of nodes will be partitioned into sets of nodes that are connected to each other (see Figure 4.1). These sets are called components. Within a component, the arcs can form cycles. Self-loops are treated as cycles of length 1. A component can have zero, one, or more than one cycle (components A, B, and C respectively in Figure 4.1). It is possible to show that if a component has more than one cycle, there is at least one arc that is the weighted sum of the other arcs in the component. Therefore, for a set of arcs to form a basis, it is necessary that no component formed by the arcs have more than one cycle. It is also possible to show that if a component has no cycle, some other component must have more than one cycle. Therefore, it is

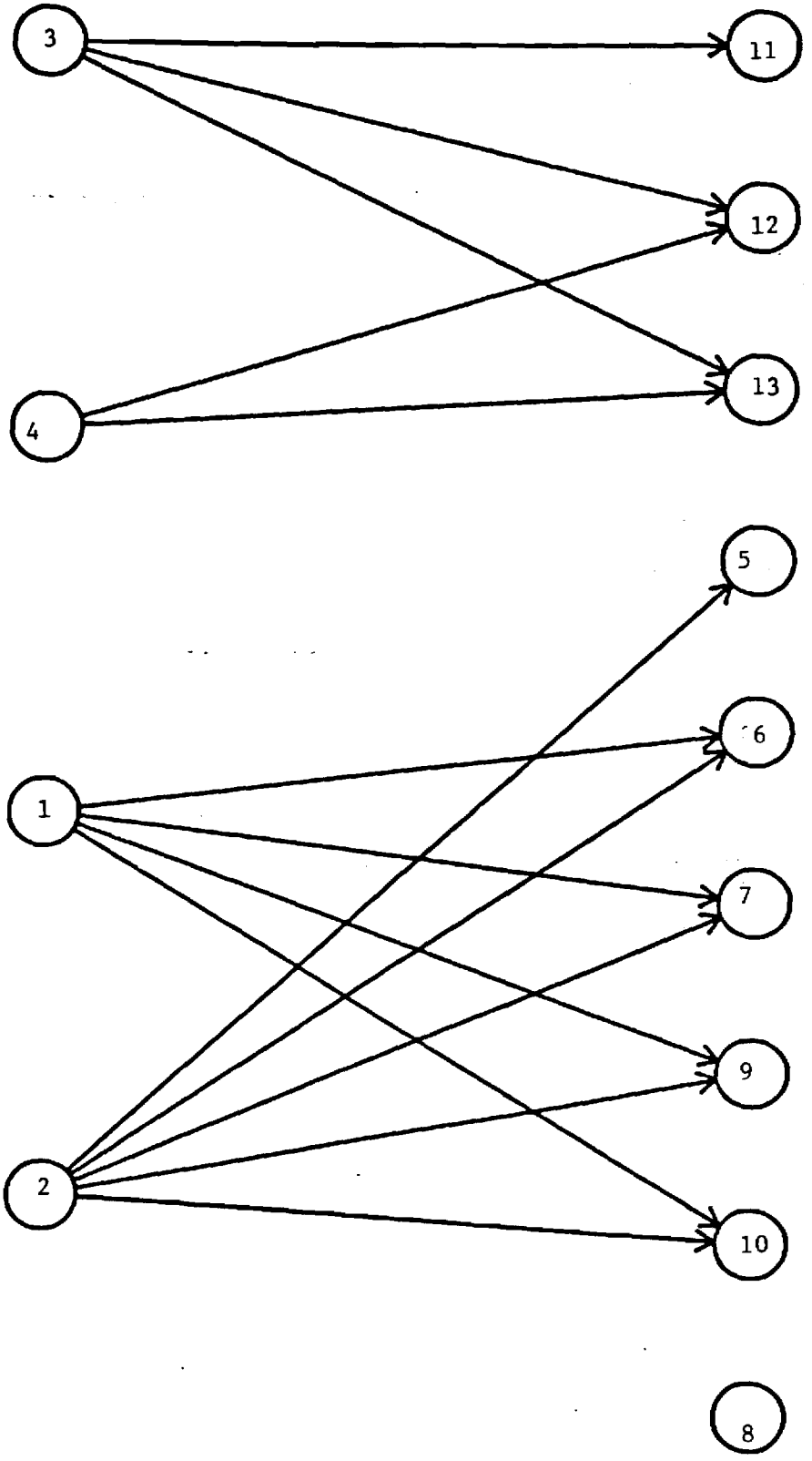


Figure 4-1. Components

also necessary that every component have at least one cycle. So every component has exactly one cycle. A component with exactly one cycle is called a "one-tree".

One further condition is required to ensure linear independence. If the cycle is not a self-loop, it is possible for an arc in the cycle to be a weighted sum of the other arcs in the cycle. A necessary and sufficient condition for this not to occur is for the cycle to have a cycle multiplier not equal to one. The cycle multiplier is calculated as follows: Assign an orientation to the cycle (clockwise or counter-clockwise). The cycle multiplier is the product of the arc multipliers for those arcs pointed in the same direction as the orientation, divided by the product of the arc multipliers of those arcs pointed in the reverse direction as the orientation.

To summarize, a set of arcs is a basis if the following conditions are satisfied:

1) The number of arcs is equal to the number of nodes.

2) Each component has exactly one cycle.

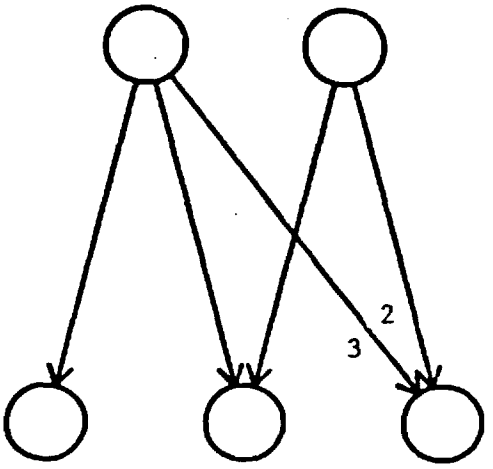
3) Each component with a cycle that is not a self-loop has a cycle multiplier that is not equal to one.

Some valid and invalid basis examples are given in Figure 4-2.

Since every multiplier in a pure network is 1, it is not possible for a pure network basis to have a cycle that is not a self loop.

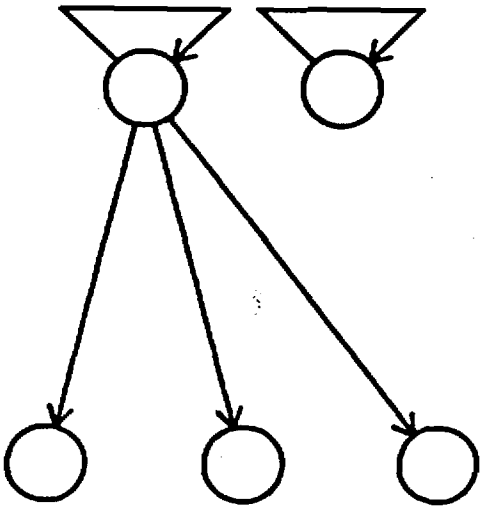
4.1.1 Specialization for the SCOPE Model

The arc multipliers in the MRMATE model have a special structure. Since every arc connects a movement requirement to either an air or a

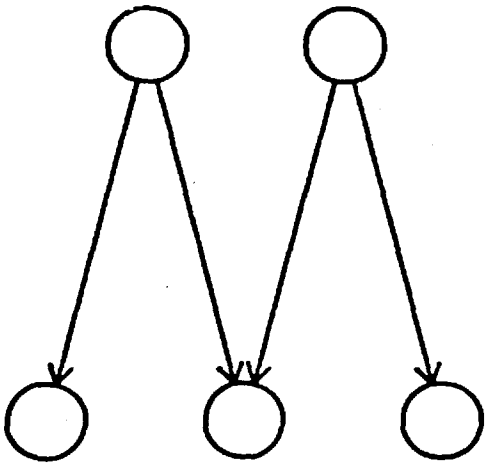


label: multiplier

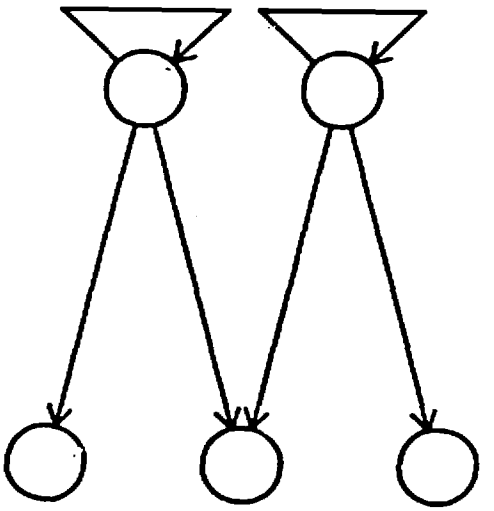
(a) valid



(b) valid

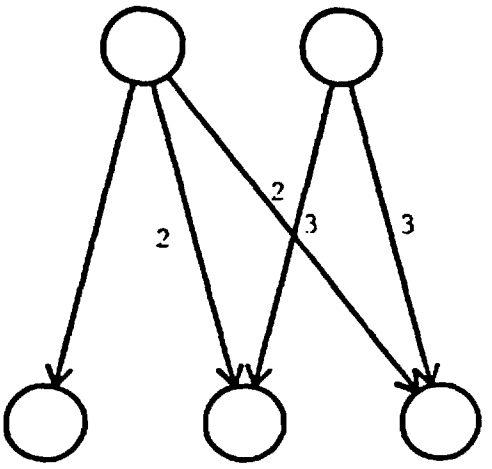


(c) invalid (no cycle)



(d) invalid (two cycles)

Figure 4-2. Valid and Invalid Basis



label on arc: multiplier

(e) invalid: cycle multiplier of 1

Figure 4-2. Continued

sea channel, the arcs have multipliers of either 1 or the conversion factor for the movement requirement. Given the way that cycles form in the MRMATE model, it is easy to show that if a component has just air channel arcs or just sea channel arcs, then the cycle associated with the component must be a self loop. In other words, if a component has only one type of arc then that component has the same basis structure as a pure network basis. Since pure networks can be solved more efficiently than generalized networks, it is likely that some advantage can be taken of the basis structure in this case.

4.2 Storing the Basis

Since the basis for generalized networks has special structure, it should be possible to store the basis in an efficient way. There are two important factors in storing the basis: storage space and computation time.

The following sections outline a method of storing the basis, called the linked rooted tree method. This method is similar to that of Brown and McBride [2], but differs in some important ways.

The linked rooted tree method is based on the data structures used for pure networks (see Kennington [5]). In this method, nodes on the cycle are seen as roots for trees consisting of nodes not on the cycle. These trees are then linked around the cycle (see Figure 4-3). Each component contains one or more trees together with the linking cycle.

4.2.1 Predecessor Structure

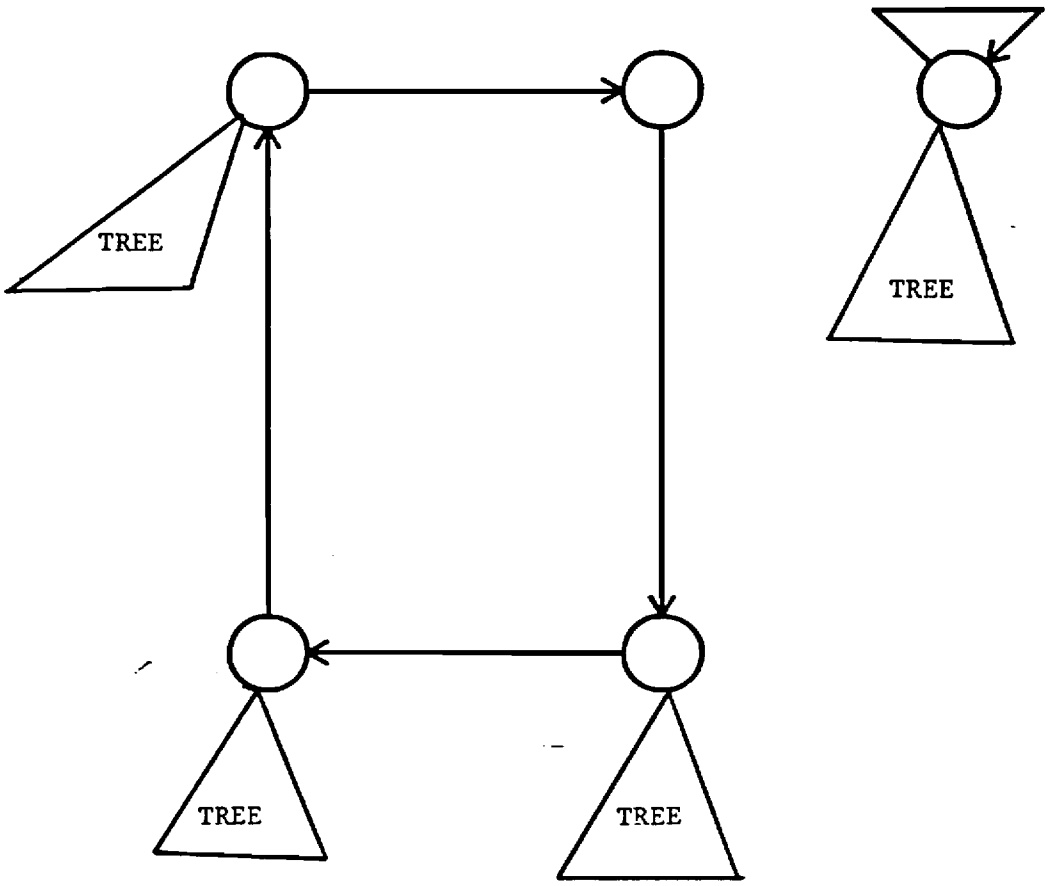


Figure 4-3. Linked Rooted Trees

The most fundamental operation required for the manipulation is to "go up" the tree. (Here the cycle is considered "on top" of the tree). This is required in determining the arc to exit the basis, for the arcs that must be checked are exactly those above the endpoints of the entering arc. It is also very useful when determining the new basis (Section 7).

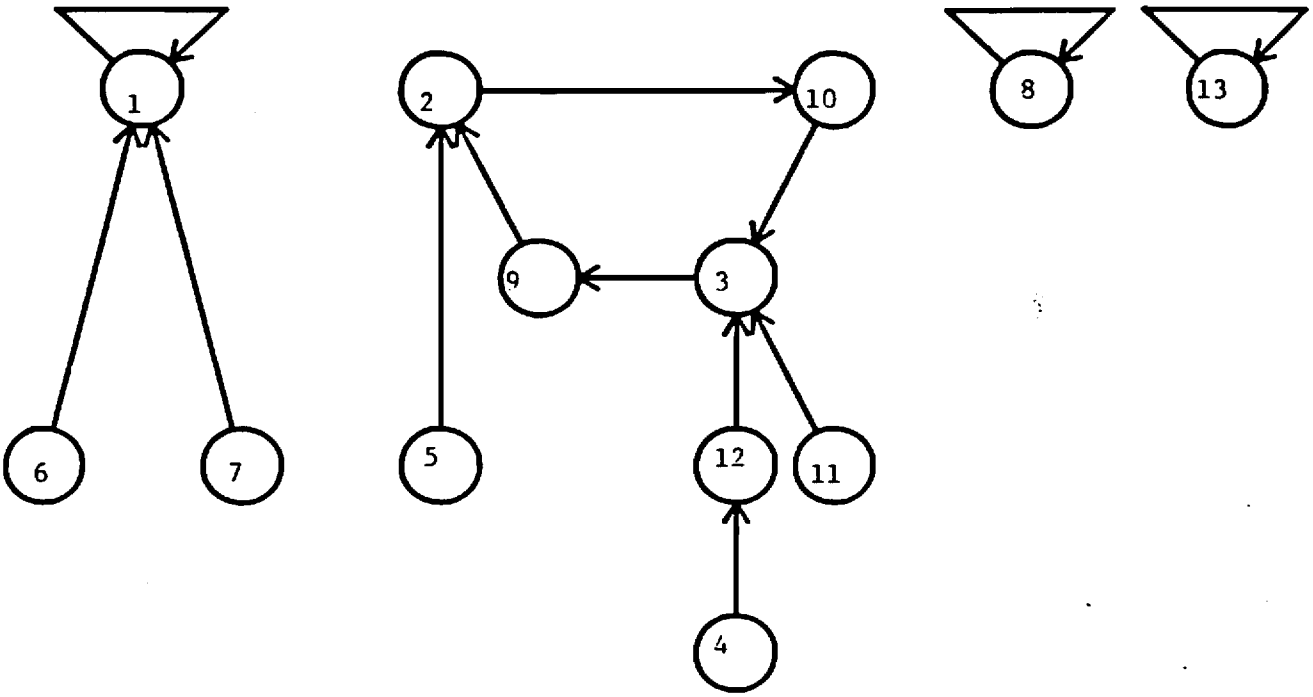
For any selected node not on the cycle, the predecessor (PRED) is defined to be the (unique) node, such that there is a basic arc connecting the two nodes and PRED is closer to the cycle than the selected node.

For nodes on the cycle, an arbitrary orientation of the cycle is selected. The PRED of a node on the cycle is the node just before it on the cycle using the selected orientation. The PRED of a node on the cycle is also on the cycle. If the cycle is a self-loop, the PRED of the cycle node is defined to be itself. (See Figure 4-4).

The PRED provides the only method of moving from one cycle node to another under the linked rooted tree system. Also note that there is no connections between components, for that ability is not required for the simplex calculations.

4.2.2 Thread Structure

The thread structure (THREAD) provides a mechanism for visiting every node in a tree. The order in which the nodes are visited is defined to be the "preorder traversal" (see Kennington [5]). This order has the property that if node X is on the path from Y to the root then node X is visited before node Y. Note that the THREAD is only within trees, not between them. (See Figure 4-5).



Node	Pred
1	1
2	10
3	9
4	12
5	2
6	1
7	1
8	8
9	2
10	3
11	3
12	3
13	13

Figure 4-4. Predecessor Structure

The THREAD is required by the basis update routines to determine those nodes whose duals and LEVELs (Section 4.2.3) must be updated.

It is in this structure that the linked rooted tree method differs from that used by Brown and McBride. In that report, the THREAD was defined traverse around the cycle in the opposite direction of PRED.

4.2.3 Level Structure

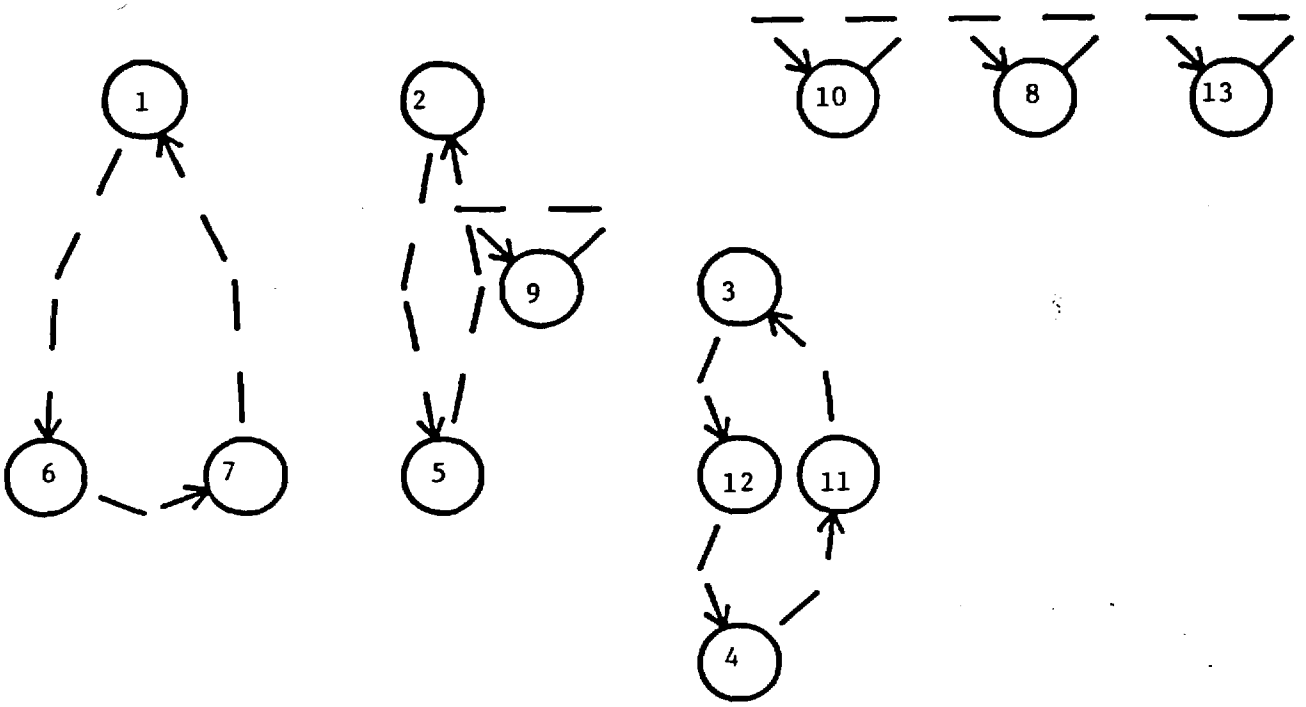
The level structure (LEVEL) gives the "distance" of a node to the cycle. Nodes on the cycle have a LEVEL value of zero. This structure is required by the routine to find the exiting arc (Section 6). (See Figure 4-6).

4.2.4 Reverse Thread Structure

The reverse thread structure (RTHREAD) is simply the inverse of the THREAD structure. This permits the visiting of nodes in reverse order. Typically, this structure is only used to make the basis update more efficient. (See Figure 4-7)

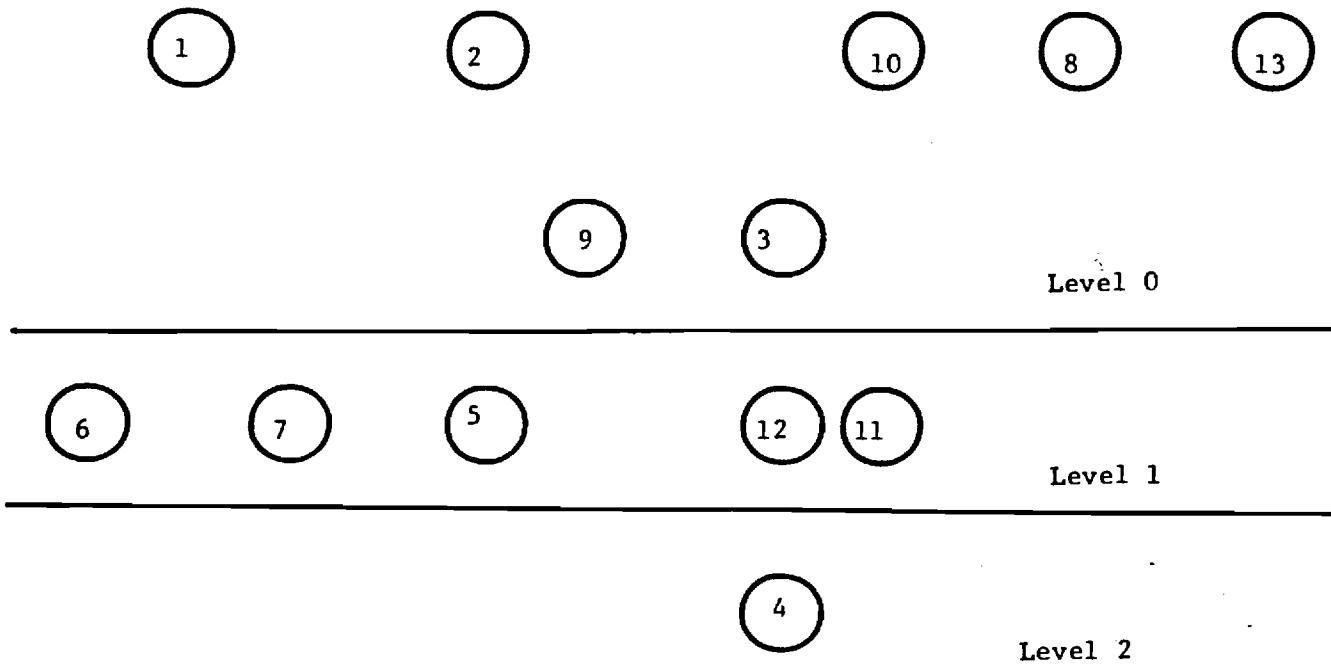
4.2.5 Arc Information Structures

Some information on the basic arcs is required to perform the simplex calculations. This includes the CAPACITY, arc multiplier (MULT) and current FLOW, to determine the arc to exit the basis; the arc COST, to update the dual variables; and the arc number (ARC) to record the optimal solution. If the information on all arcs (basic



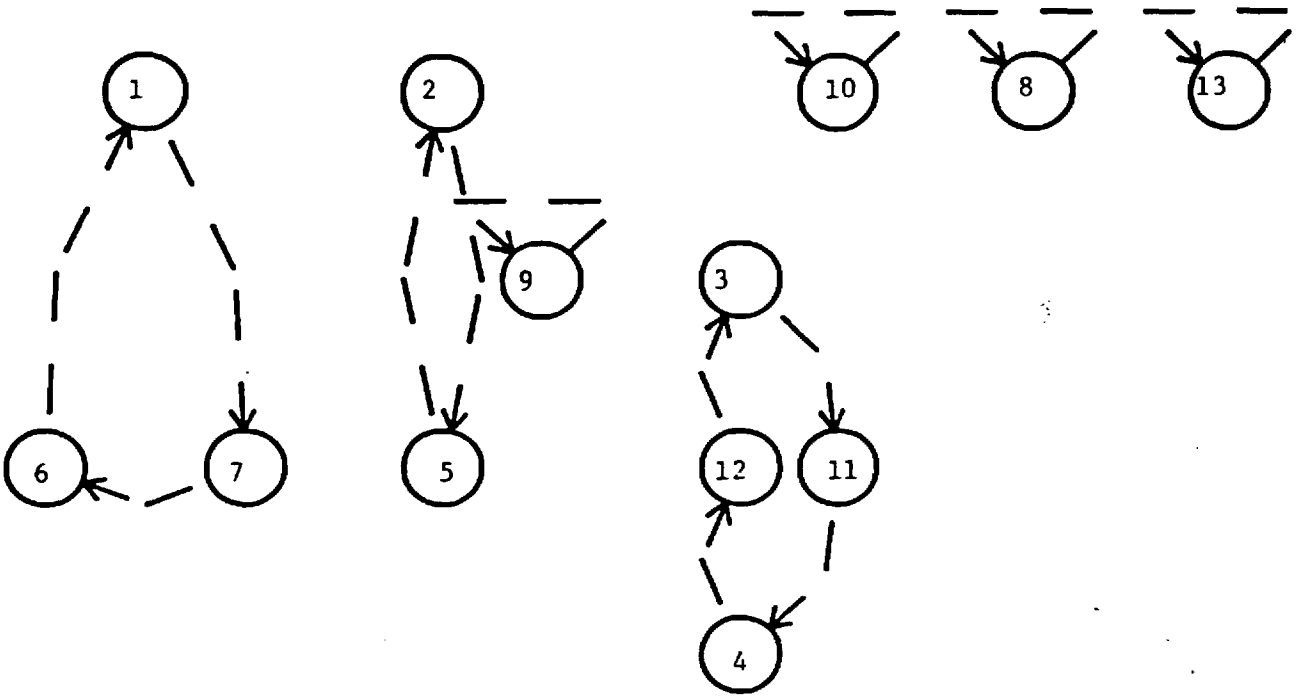
Node	Thread
1	6
2	5
3	12
4	11
5	2
6	7
7	1
8	8
9	9
10	10
11	3
12	4
13	13

Figure 4-5. Thread Structure



Node	Level
1	0
2	0
3	0
4	2
5	1
6	1
7	1
8	0
9	0
10	0
11	1
12	1
13	0

Figure 4-6. Level Structure



Node	Reverse Thread
1	7
2	5
3	11
4	12
5	2
6	1
7	6
8	8
9	9
10	10
11	4
12	3
13	13

Figure 4-7. Reverse Thread Structure

and non-basic) is available, then it is only necessary to store the ARC value explicitly. As Section 5.1 will show, however, for large problems it is necessary to have only a limited amount of arc information available at any given time. Therefore, all of the above basic arc information must be stored.

The information on the basic arc that connects NODE and PRED(NODE) is associated with NODE. Since it is not clear whether the arc begins at NODE and ends at PRED(NODE) or the reverse, the ARC value is given a sign depending on the orientation. In the former case, ARC is positive; in the latter, ARC is negative. (See Figure 4-8).

4.2.6 Dual Value Structure

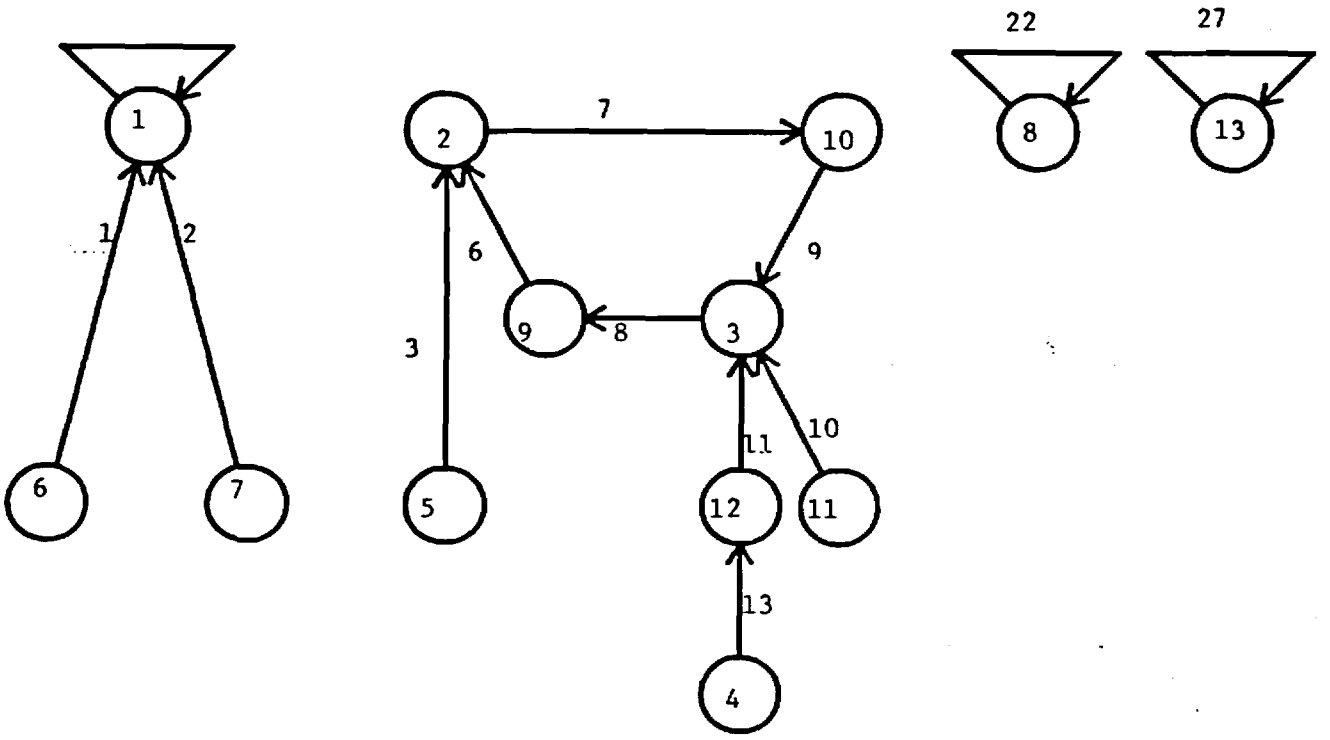
The dual variable (DUAL) for each node is required to determine an arc to enter the basis. Therefore, the dual for each node is retained at all times.

4.2.7 Cycle Multiplier

The cycle multiplier (CMULT) is defined as follows: give an orientation to the cycle; the cycle multiplier is the product of the arc multipliers for arcs in the same direction as the orientation divided by the arc multipliers of those arcs in the reverse direction.

CMULT is defined only for nodes on cycles that are not self-loops. CMULT is the same for all nodes on the same cycle.

The cycle multiplier can be thought of as the amount of flow that



label on arc: arc number
direction on arc: predecessor

Node	Arc
1	15
2	7
3	8
4	13
5	-3
6	-1
7	-2
8	22
9	-6
10	-9
11	-10
12	-11
13	-27

Figure 4-8. Arc Structure

will result if one unit of flow is sent around the cycle in the given orientation. In a valid basis, CMULT cannot be one.

CMULT is needed to determine the arc to exit the basis, and to update the flows. Because of the number of multiplications and divisions required, this is a very time-consuming number to calculate. Fortunately, this number must be calculated just once for each new cycle created. So pivots that do not create a new cycle do not require the calculation of any CMULT values.

5. Handling Arcs

In general there are far more arcs than nodes in a network model. The handling of the arcs is therefore critical to decreasing computation time and storage requirements. Despite the large amount of arc data, very little data is required for any individual pivot. This permits the storage of the arc data outside central computer memory, normally on a high speed mass storage device.

Methods for handling the arc data was reported in detail in [84-09]. The following sections review the conclusions of that report.

5.1 Choosing an Entering Arc

The primal simplex method provides flexibility in the choice of arc to enter the basis. The only property the entering arc must have is that placing a small amount of flow on the arc will decrease the objective function value. It is not necessary to select the arc that will yield the greatest decrease in the objective value the most.

Given the current dual values, it is easy to determine the effect of making a small change in the current flow of a non-basic arc. If the arc currently has no flow on it, increasing the flow by one unit would add the following amount to the objective function:

MULT * DUAL(HEAD) - DUAL(TAIL) - COST

where the arc in question is from node TAIL to node HEAD and has multiplier MULT and cost COST.

If the current flow on the arc is the capacity of the arc, then decreasing the flow by one unit would add the following amount to the

objective:

- (MULT * DUAL(HEAD) - DUAL(TAIL) - COST)

It might not be possible to change the flows by one unit. Some arc in the basis could reach one of its bounds before one unit of flow is placed on the entering arc. Conversely, it might be possible to change the flow by more than one unit. The actual amount of change is determined in Step 3 of the primal simplex algorithm.

The above equations are referred to as the reduced cost for the arc. If the reduced cost is negative, the objective value would be decreased if the non-basic arc were to enter the basis. Non-basic arcs with negative reduced cost are eligible to enter the basis. If there is no arc with a negative reduced cost then the current solution is optimal.

Generally, there are many arcs that are eligible to enter the basis at each iteration. It is necessary to choose from among those possible. For instance, it is possible to calculate all of the reduced costs and choose the arc with the most negative reduced cost to enter the basis. This, generally, will have fewer pivots than other methods but the amount of time required to calculate all of the reduced costs would be prohibitive.

Another alternative is to calculate the reduced costs, one at a time, and choose an arc to enter the basis as soon as one is found with a negative reduced cost. This will minimize the amount of time to calculate reduced costs; but it will cause many pivots which improve, only marginally, the objective value.

In practice, two methods are used to choose an arc to enter the basis. These methods are referred to as the fixed page method and the candidate list method.

5.1.1 Fixed Page Method

In the Fixed Page method, the reduced cost for a fixed number of arcs (a page) is calculated at each pivot. The arc with the most negative reduced cost is then selected to enter the basis. If no arc has a negative reduced cost, then a new page (of arcs) is used. The page size (number of arcs in the page) is important. Too small a page will cause too many pivots; too large a page will cause too much time for the reduced cost calculation.

After each pivot, a decision must be made whether to use the same page of data or to obtain a new page. One method for making this decision is to provide a re-use factor, giving the maximum number of times a page can be used before a new page must be selected.

5.1.2 Candidate List Methods

If arcs from a variety of nodes are examined, then, when one of them enters the basis, generally, it will not effect the reduced cost of many of the other arcs. It is therefore possible to examine only a subset of arcs, called a candidate list. The first step is to create a list of arcs with a negative reduced cost. The arc with the most negative reduced cost is then selected to enter the basis. The reduced costs of the arcs in the list are then recalculated, and the next arc to enter the basis is selected from the list. After a fixed number of pivots, the candidate list is reformed.

Two parameters are required: the candidate list size and the number of iterations before reforming the list.

5.2 Storing Arc Data

In both major methods for choosing an entering arc, only a small amount of arc data is required at any given time. Information on arcs in the basis is always required, but only a small number of non-basic arcs are needed. In the fixed page method, only arcs in the page being examined are needed. In the candidate list method, only arcs in the candidate list are required.

This suggests maintaining arc data on a high speed mass storage device (e.g. hard disk on a microcomputer). Only basic arc information and a small number of search pages is maintained in core. When new arc data is required a page of arcs can be read in, replacing the previous pages. The amount of data remains constant while the actual data is constantly changing. This is called the "in-core/out-of-core" method.

Some method is needed to store information on the non-basic flows. The non-basic flows are either zero or the arc's upper bound. The various possibilities were given in [3]. If there is a large number of arcs then the information on the non-basic flows must also be stored outside central memory. This is slow, but it permits even small computers to solve extremely large problems.

5.3 Specialization for MRMATE

The number of arcs in a MRMATE model can be very large. The largest problems can have more than one-half of a million arcs. Problems of this size require the in-core/out-of-core method.

One advantage of the MRMATE problem is that the structure of the arc costs is known. There will be many arcs with zero cost. These arcs are likely to be in the optimal basis. Therefore, it seems reasonable to enter zero cost arcs as often as possible.

The remaining arcs can be separated into two classes: low cost and high cost arcs. By separating the arcs into three different files (zero, low, and high cost files) arcs with zero cost can be preferentially entered, without calculating reduced costs for low and high cost arcs. To ensure optimality all the arcs must be examined; however, more time can be spent with the zero cost arcs.

Slack and surplus arcs are also very important in the solution process. These arcs should be examined more often than other arcs. If the "wrong" slack and surplus arcs are in the basis many pivots might be performed unnecessarily. These arcs should not be kept out of core. Information necessary to generate these arcs should be available in core and their reduced costs should be recalculated frequently.

6. Finding the Exiting Arc

In the simplex method specialized for generalized networks, only a limited number of arcs are candidates to exit the basis during any pivot. The rapid identification of these arcs is a reason the specialized method is more efficient than the simplex method for general linear programming.

For any node, NODE, define the "backpath" of NODE to be those arcs between NODE and the cycle for the component that contains NODE, as well as those arcs on the cycle. In other words, the backpath for NODE contains those basic arcs whose corresponding node can be reached from NODE by use of the PRED structure only. For the entering arc number ARC, the only arcs to change flow are those on the backpaths of TAIL(ARC) and HEAD(ARC). Figure 6-1 shows the arcs that will change flow for various combinations of HEAD(ARC) and TAIL(ARC), referred to as HEAD and TAIL respectively.

If a single unit of flow were placed on the arc entering the basis, the flows on the arcs in the backpaths are the only ones which must be updated so as to keep the net flow at each node the same. For instance, the arc between TAIL and PRED(TAIL) must provide one unit of flow at node TAIL. If that arc is oriented from PRED(TAIL) to TAIL and has multiplier of MULT, then the arc between PRED(PRED(TAIL)) and PRED(TAIL) must provide 1/MULT units of flow at PRED(TAIL), and so on (see Figure 6-1).

This calculation is equivalent to determining the updated column in linear programming. With this updated column, it is possible to determine the amount of flow by which the entering arc flow can

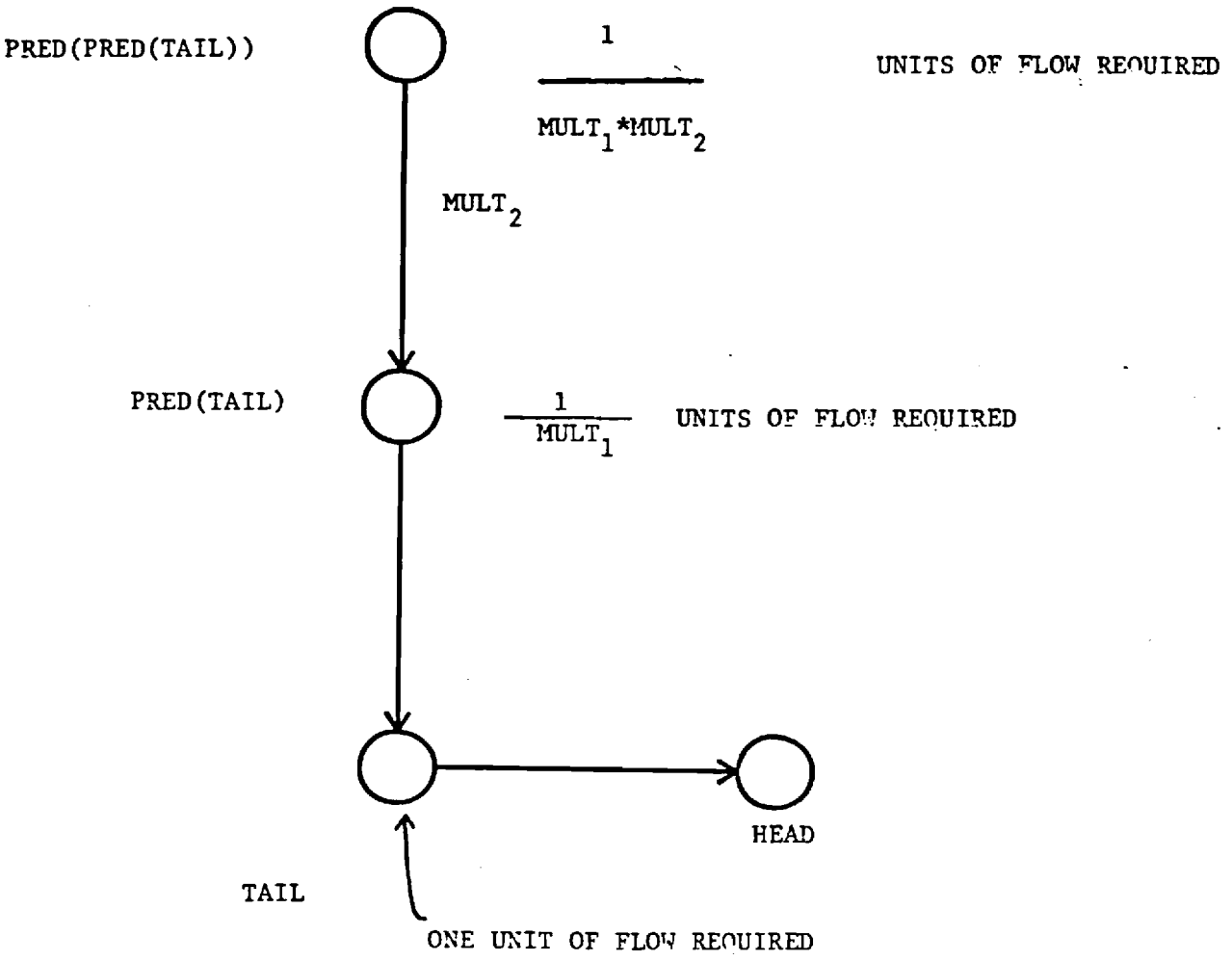


Figure 6-1. Flow Required at Nodes

change before some basic arc reaches one of its bounds. The first arc to reach one of its bounds is the exiting arc from the basis.

6.1 Calculating the Exiting Arc

The determination of the exiting arc involves two steps: the development of the updated column and the determination of the maximum flow change on the entering arc, ARC (before a basic arc reaches one of its bounds). These two steps can be performed simultaneously for each node. For simplification, the presentation here will separate the process. There are several cases to consider.

The first case assumes that the backpaths from TAIL and HEAD are distinct, implying that TAIL and HEAD are in different components. If the multiplier on ARC is MULT then placing one unit of flow on ARC will require one unit of flow at TAIL and will provide MULT units of flow at HEAD. The sign convention of a negative number for a demand and a positive number for a supply will be adopted. The value of -1 for TAIL and MULT for HEAD is called the requirement.

Given the requirement at a node NODE, two pieces of information are required: the update column entry for the arc between NODE and PRED(NODE) and the requirement for PRED(NODE).

These values depend on the orientation of ARC(NODE), the basic arc between NODE and PRED(NODE). The following algorithm calculates the entry in the updated column (UP_COL) and updates the requirement for NODE (REQUIRE) to be the requirement for PRED(NODE).

```
if ARC(NODE) < 0 then (*arc oriented from PRED(NODE) to NODE *)
     REQUIRE := REQUIRE/MULT(NODE);
     UP_COL(NODE) := REQUIRE;
else
     UP_COL(NODE) := -REQUIRE;
```

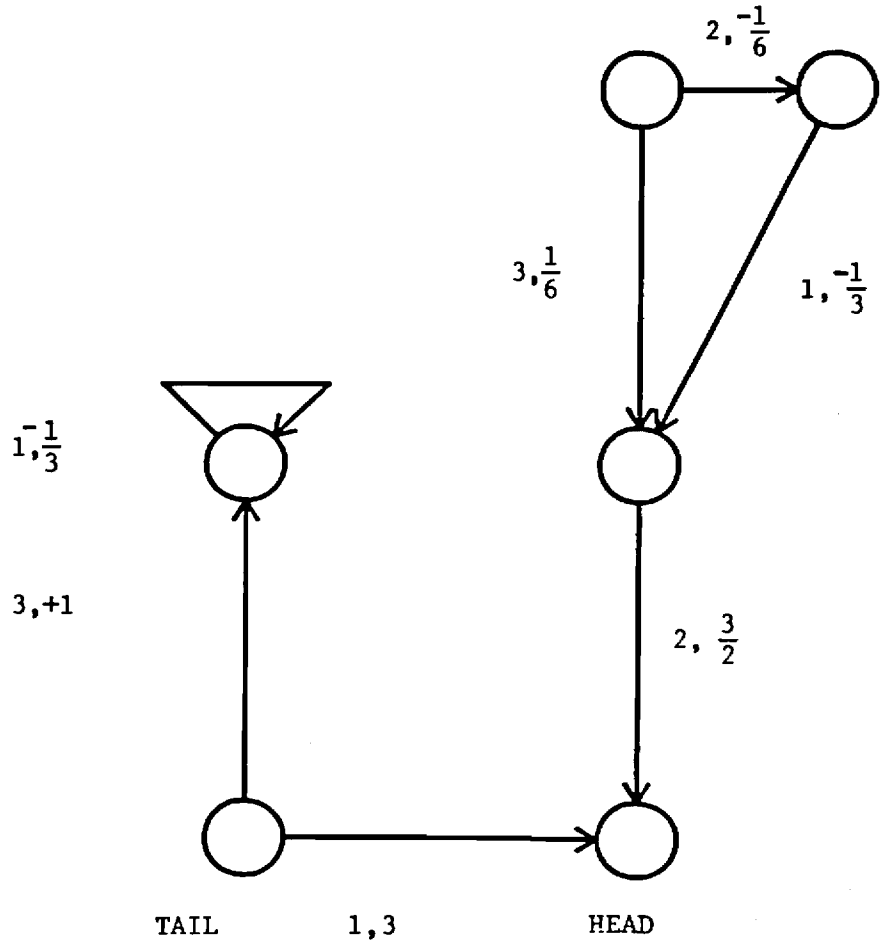
```
        REQUIRE := REQUIRE * MULT(NODE);
    endif
```

The process is slightly more complicated when a cycle, not a self-loop, is reached. The effect of the cycle multiplier must be taken into effect. Briefly, the cycle is used to create or destroy flow as needed. The CMULT value determines the rate at which flow can be created or destroyed. The effect of the cycle multiplier is that when the cycle is reached, REQUIRE is replaced by REQUIRE / (1-CMULT(CYCLE)). The equations above can then be used for the arcs on the cycle. These calculations can be carried out independently for the backpaths of HEAD and TAIL when the nodes are in different components.

For the second case, when the two nodes are in the same component, the backpaths will coincide at some point. The updated column is the sum of the updated columns calculated using the above equations. If the backpaths coincide before the root cycle it is possible to simply add together the REQUIRE values of the two backpaths and continue as though only one backpath existed. If the value of REQUIRE is zero (as it will be for pure networks) no further calculations need be done; the rest of the arcs will not change in flow. If the backpaths coincide only on the cycle, it is easiest to proceed around the cycle twice making the necessary calculations and add together the resulting updated columns.

An example of these calculations is given in Figure 6-2.

Given UP_COL(NODE) it is possible to determine the amount of change permitted on the entering arc before the arc associated with NODE reaches a bound. Let INCREASE := 1 if the entering arc is



label: MULT, UP_COL

Figure 6-2. Updated Column

currently at zero flow and -1 if the entering arc has flow equal to its capacity. The calculation of the maximum change is as follows:

```
if (INCREASE * UP_COL(NODE) > 0) then
     (* flow will decrease on ARC(NODE) *)
     MAX := FLOW(NODE) / UP_COL(NODE);
else
     (* flow will increase on ARC(NODE) *)
     MAX := (CAPACITY(NODE) - FLOW(NODE)) / UP_COL(NODE);
endif
```

By taking the minimum value for MAX over all nodes with a changing flow, the flow change on the entering arc is determined and the exiting arc is identified. If this value is more than the CAPACITY of the entering arc then the change in flow is the CAPACITY of the entering arc and the exiting arc is the entering arc.

7. Updating the Basis

Given the arc to enter and and the arc to leave the basis, the final step is to update the basis structures. This update requires updating the DUALs, the FLOWs and the rest of the basis structures.

7.1 Updating the Basis Structures

7.1.1 Pivot Types

The linked rooted tree method has six pivot types, depending on the relationship between the entering and exiting arc.

Pivot type 1 occurs when the entering arc and exiting arc are the same. This occurs when flow on the entering arc reaches its upper bound or zero before any other flows reach their limits. In this case, the basis remains the same, so only FLOWs must be changed.

When TAIL and HEAD of the entering arc are in the same tree, the pivot type is defined to be either 2, 3 or 4. Consider paths from the entering tail node to the cycle and from the entering head node to the cycle. The first node that occurs on both paths is called the meeting node (MEET). The (common) cycle node is called CYC. The exiting arc can occur in three places: before MEET, between MEET and CYC, and after CYC. These three places correspond to pivot types 2, 3 and 4 respectively.

When TAIL and HEAD of the entering arc are in the same component, the pivot type is 2 or 5. If the exiting arc is on the cycle, the pivot type is 5. Otherwise it is type 2.

If TAIL and HEAD of the entering arc are in different components, the pivot type is 6 if the exiting arc is on a cycle, and is 2 otherwise.

Figure 7-1 gives examples of all of these pivot types.

7.1.2 Common Routines

The main advantage of the linked rooted tree method of storing the generalized network basis is that the basis update routines involve a limited number of tree manipulation routines. Each pivot type uses these routines in a different way to create the new basis.

There are five tree manipulation routines required:

1) Hang tree (HANG) : Takes two trees and a node within the first tree and attaches the second tree to the first below the node.

2) Isolate subtree (ISOLATE): Takes a tree and a node within the tree and isolates the subtree below the node by creating a new tree.

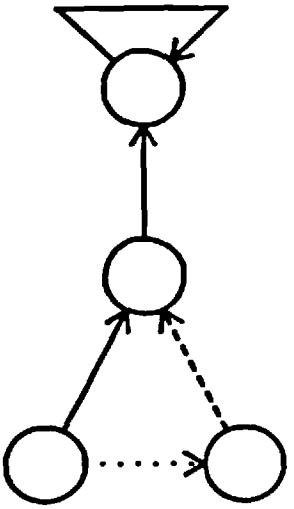
3) Reroot tree (REROOT) : Takes a tree and a node within the tree and makes that node the root of the tree.

4) Reverse cycle reroot (REV_CYC_REROOT): Takes a series of trees connected by PRED values and creates a new tree consisting of all of them in reverse order.

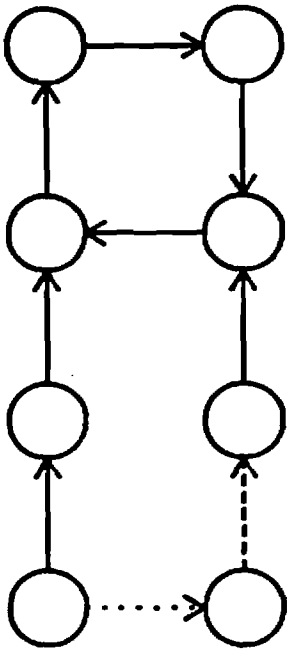
5) Cycle reroot (CYC_REROOT): Takes a series of trees connected by PRED values and creates a new tree consisting of all of them.

7.1.2.1 HANG Routine

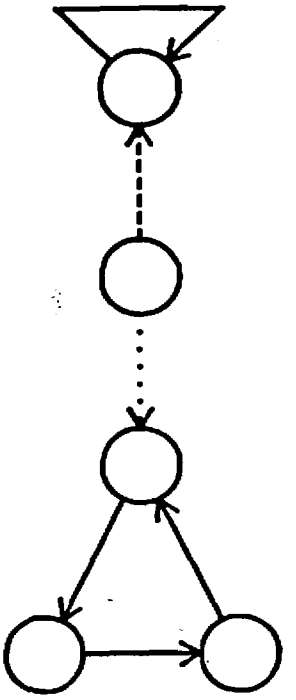
INPUT:



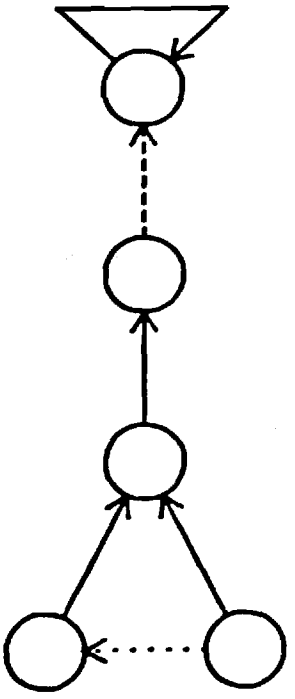
(a) pivot type 2



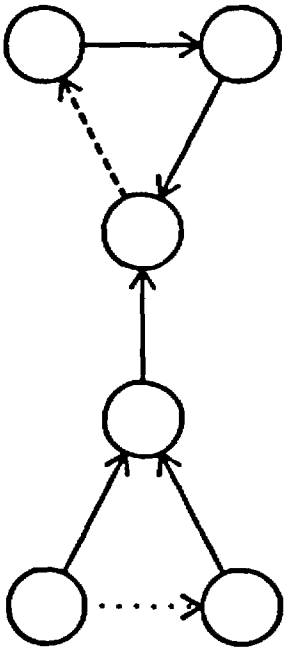
(b) pivot type 2



(c) pivot type 2



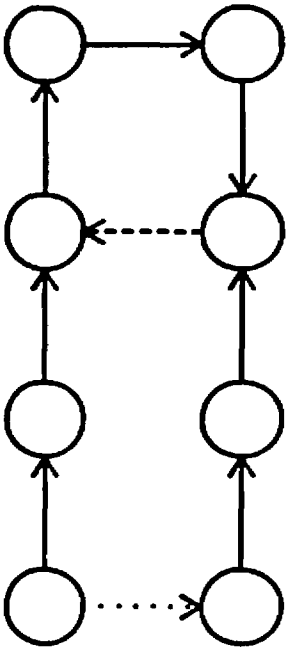
(d) pivot type 3



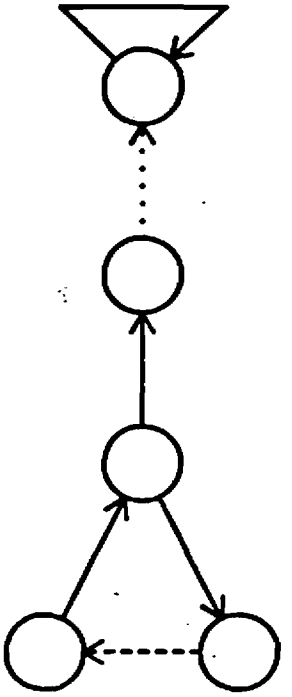
(e) pivot type 4

Key: ... entering arc
--- exiting arc
___ other basic arcs

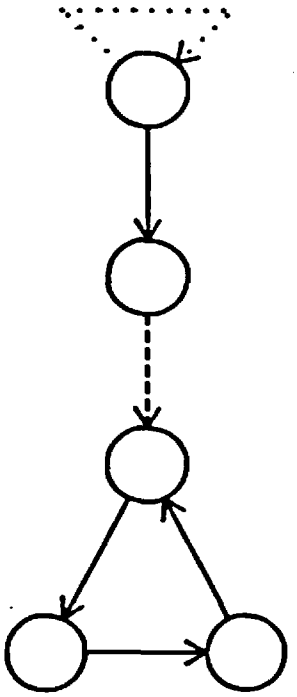
Figure 7-1. Pivot Types



(f) pivot type 5



(g) pivot type 6



(h) pivot type 7

Figure 7-3. Pivot Types (cont.)

ROOT: a node, not necessarily the root of one tree;

TREE: the root node of another tree.

OUTPUT:

Updated PRED, THREAD, LEVEL and RTHREAD for a tree with the nodes of TREE below ROOT. (See Figure 7-2).

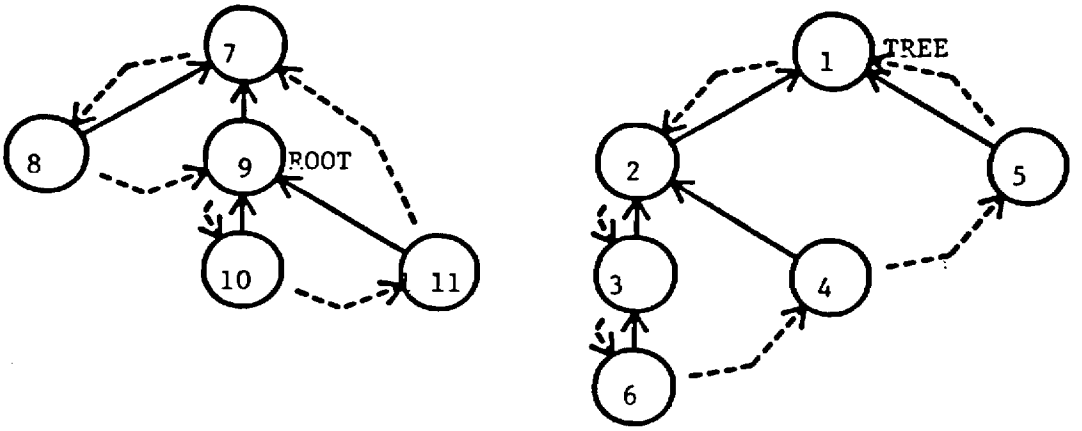
METHOD:

```
(A)  Find the final node in preorder traversal (LAST) in the
     subtree below ROOT by following the THREAD values;

(B)  LEVEL(TREE) := LEVEL(ROOT) + 1;
     NODE := THREAD(TREE);
     while NODE <> TREE do
          LEVEL(NODE) := LEVEL(PRED(NODE)) + 1;
          NODE := THREAD(NODE);
     endwhile;

(C) TEMP := THREAD(LAST);
    THREAD(LAST) := TREE;
    THREAD(RTHREAD(TREE)) := TEMP;
    RTHREAD(TEMP) := RTHREAD(TREE);
    RTHREAD(TREE) := LAST;
    PRED(TREE) := ROOT;
```

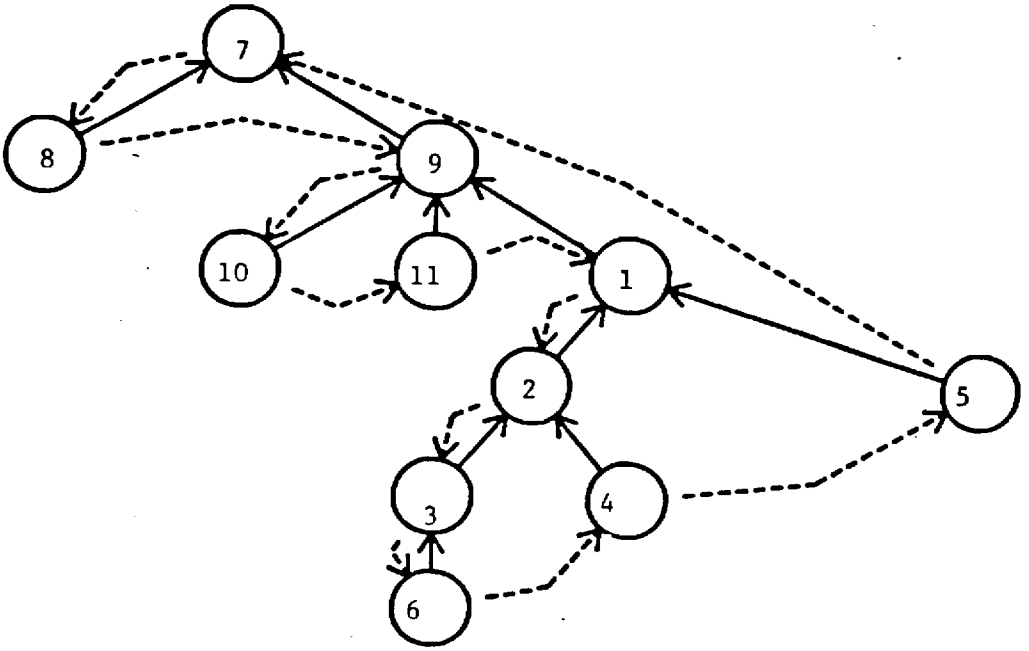
DISCUSSION:

Two common programming structures are exhibited in this routine. In section B, the LEVEL structure is updated for TREE. Because the THREAD is the preorder traversal, the level of PRED(NODE) is always calculated before the level of NODE. Since LEVEL(NODE) is always equal to LEVEL(PRED(NODE)) + 1 within trees, the level calculation is simplified.

The second structure is in section C. For roots of trees, the reverse thread of the root is always the last node in the preorder traversal. This means that finding the last node in TREE is easy, as opposed to the last node below ROOT. Section A is needed to find the



BEFORE HANG



AFTER HANG (9,1)

Figure 7-2. HANG Routine

last node below ROOT.

7.1.2.2 ISOLATE Routine

INPUT:

NEW_ROOT: a node in a tree

OUTPUT:

Updated THREAD and RTHREAD so that nodes below NEW_ROOT form a tree rooted at NEW_ROOT. (See Figure 7-3).

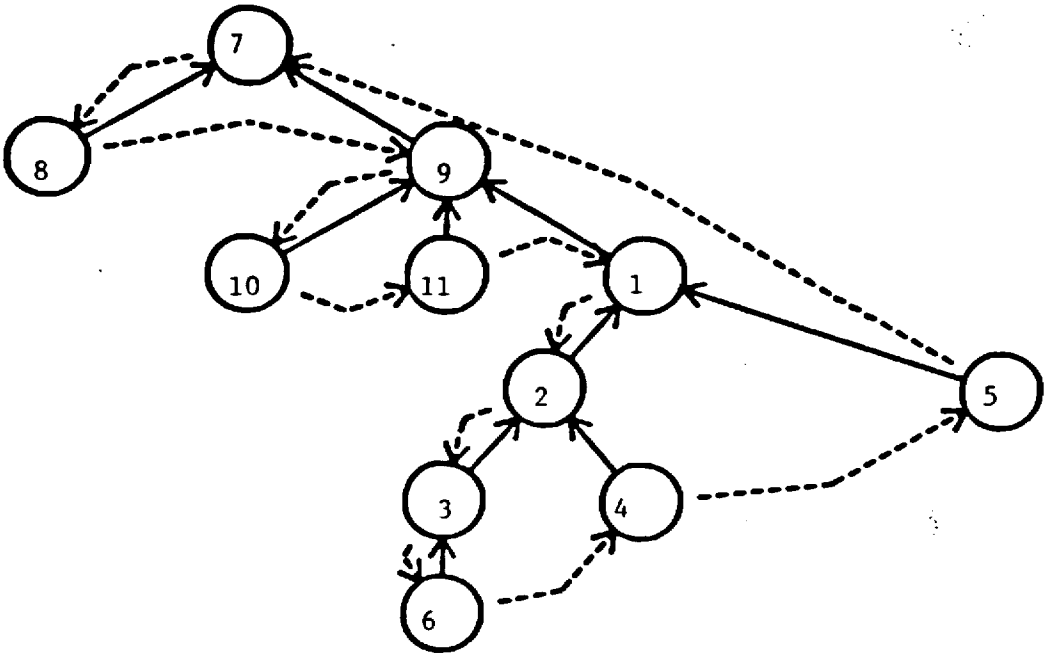
METHOD:

(A) Find the last node (LAST) in the subtree rooted at NEW_ROOT by following the THREAD values;

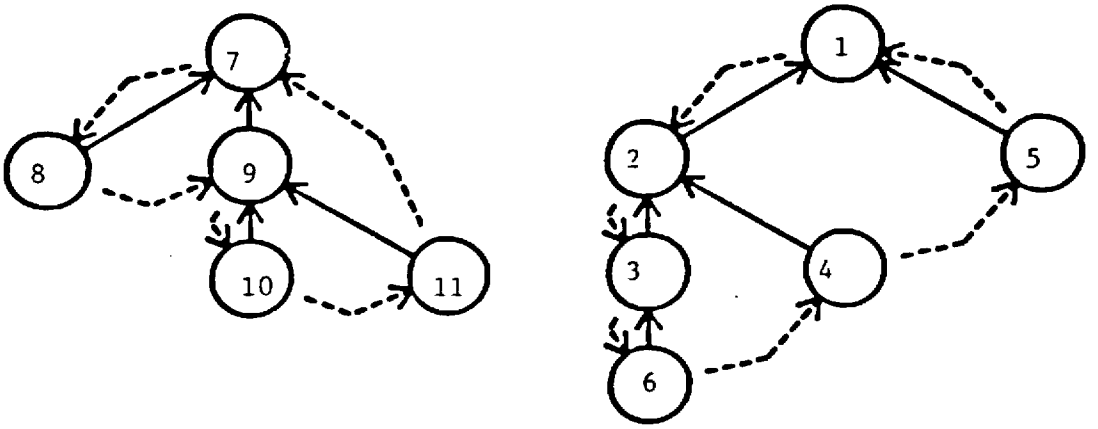
(B)
```
THREAD(RTHREAD(NEW_ROOT)) := THREAD(LAST);
RTHREAD(LAST) := RTHREAD(NEW_ROOT);
THREAD(LAST) := NEW_ROOT;
RTHREAD(NEW_ROOT) := LAST;
```

DISCUSSION:

Once the last node in the subtree below NEW_ROOT has been located, rethreading involves only nodes LAST and NEW_ROOT.



BEFORE ISOLATE



AFTER ISOLATE (1)

Figure 7-3. Isolate Routine

7.1.2.3 REROOT Routine

INPUT:

TREE: the root node of the subtree to be rerooted;

NEW_ROOT: a node in the tree rooted at TREE.

OUTPUT:

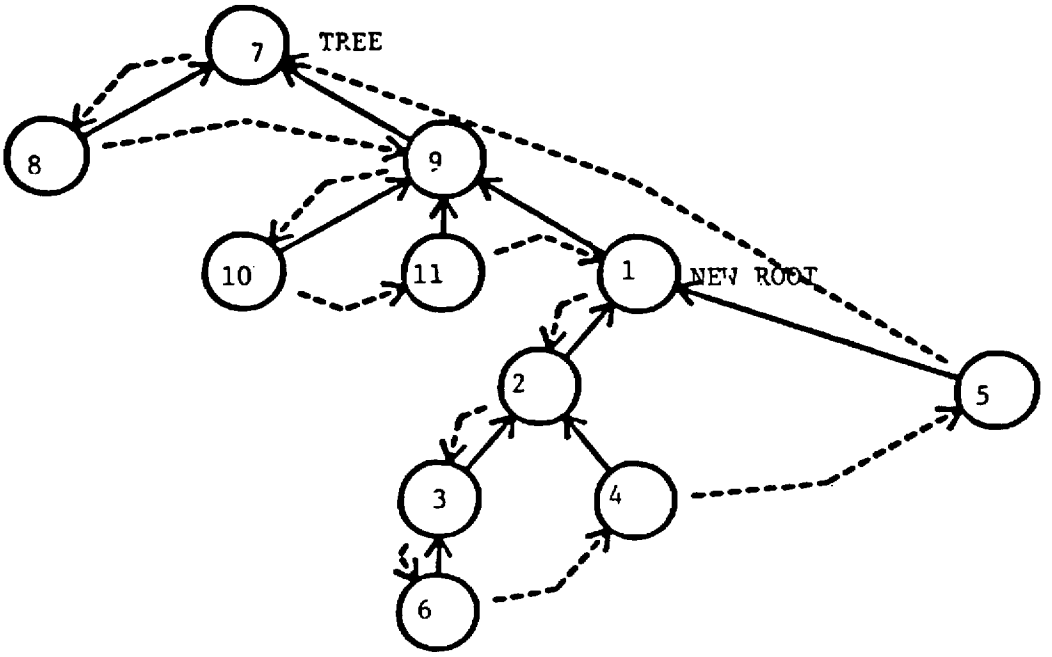
Updated PRED, THREAD, LEVEL and RTHREAD for a tree consisting of the nodes below TREE, with root, NEW-ROOT. (See Figure 7-4).

METHOD:

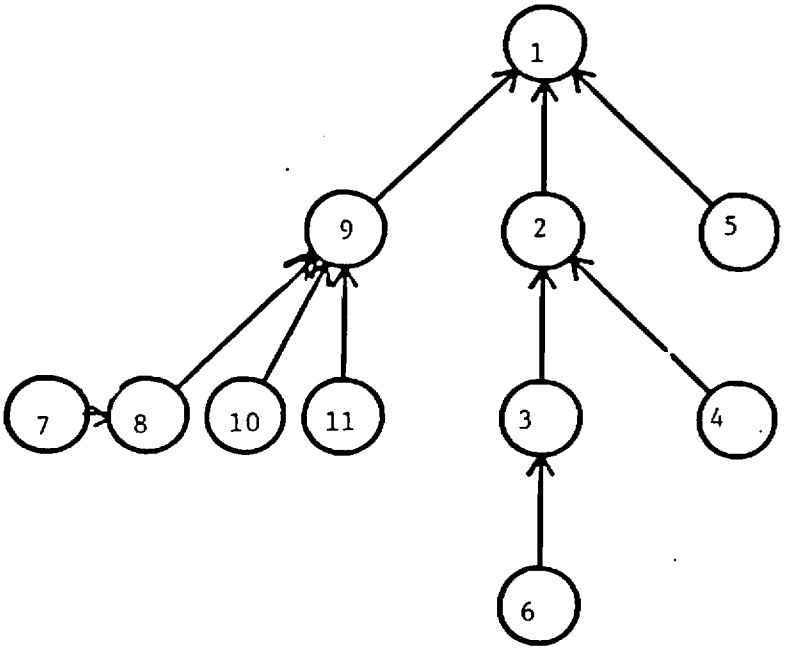
```
LEVEL(NEW_ROOT) := 0;
NODE := NEW_ROOT;
PREV := NEW_ROOT;
while PREV <> TREE do
     ISOLATE(NODE);
     TEMP := PRED(NODE);
     if NODE <> NEW_ROOT then
          HANG(PREV,NODE);
     endif;
     PREV := NODE;
     NODE := TEMP;
endwhile;
```

DISCUSSION:

Using HANG and ISOLATE, rerooting a tree at a new node is a simple task. Each node on the path from NEW_ROOT to TREE is isolated and then hung from the previous node. The temporary variable TEMP is required because HANG changes the value of PRED(NODE).



BEFORE



After Reroot (7,1)

Figure 7-4. Reroot Routine

7.1.2.4 REV_CYC_REROOT Routine

INPUTS:

FIRST_TREE: First tree of the sequence to put together;

LAST_TREE: Last tree of the sequence;

OUTPUTS:

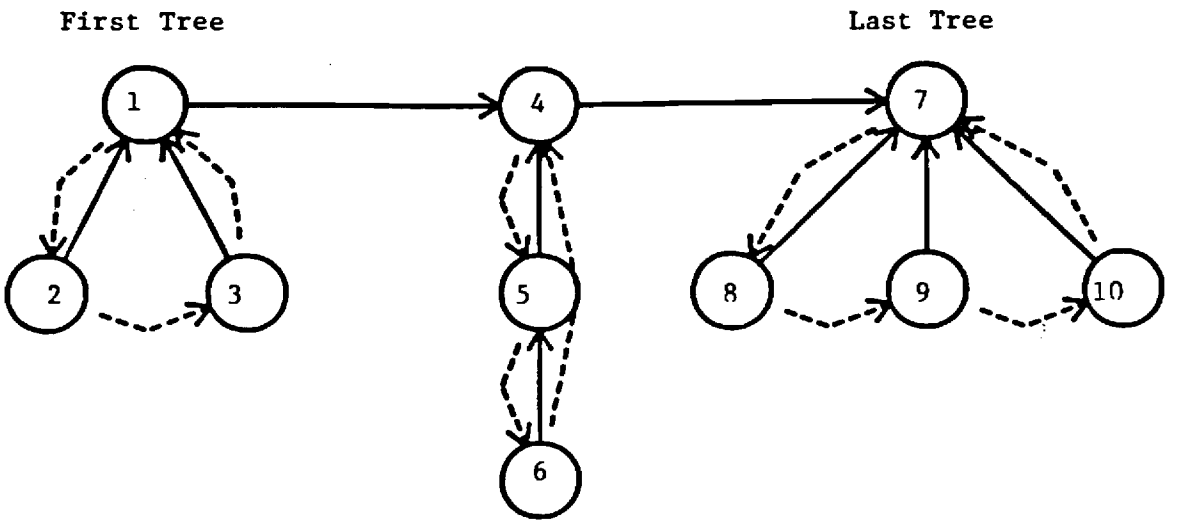
Updated PRED, THREAD, RTHREAD and LEVEL values for a single tree, rooted at FIRST_TREE, consisting of trees rooted at predecessor values from FIRST_TREE to LAST_TREE. Note that this definition requires that the predecessors along the path from FIRST_TREE to LAST_TREE be reversed. (See Figure 7-5).

METHOD:

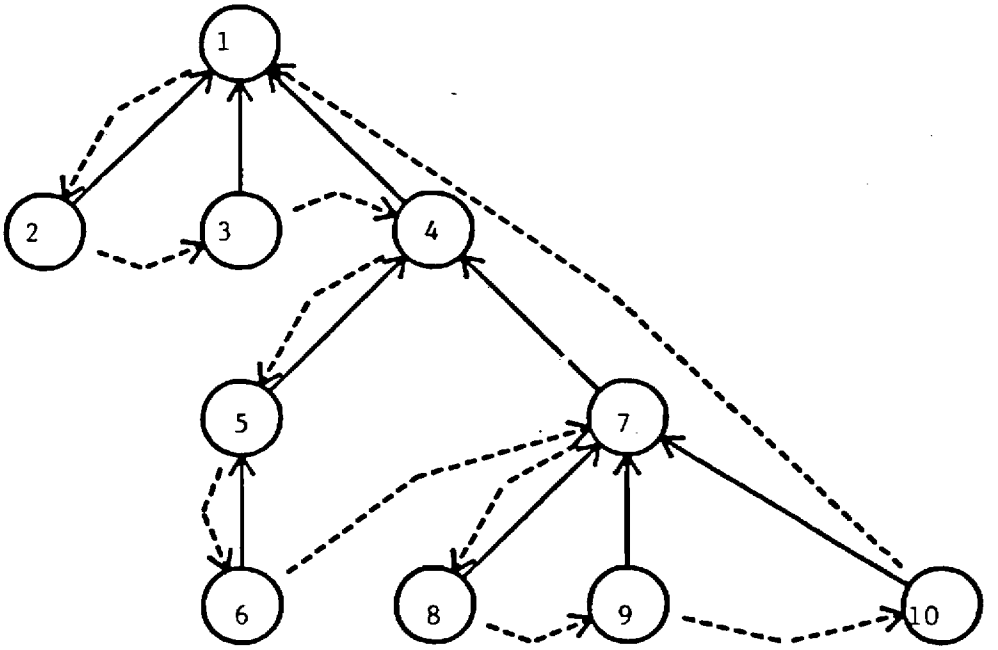
```
NODE := FIRST_TREE;
PREV := FIRST_TREE;
while PREV <> LAST_TREE do
     TEMP := PRED(NODE);
     if NODE <> FIRST_TREE do
          HANG(PREV,NODE);
     endif
     PREV := NODE;
     NODE := TEMP;
endwhile;
```

DISCUSSION:

This routine is essentially the same as REROOT, except that ISOLATE has already been accomplished.



Before



After Rev-Cyc-Reroot (1,7)

Figure 7-5. Rev-Cyc-Reroot Routine

7.1.2.5 CYC_REROOT

INPUTS:

FIRST_TREE: First tree of the sequence to put together;

LAST_TREE: Last tree of the sequence;

OUTPUTS:

Updated PRED, THREAD, RTHREAD and LEVEL values for a single tree, rooted at LAST_TREE, consisting of trees rooted at predecessor values from FIRST_TREE to LAST_TREE. This differs from REV_CYC_REROOT only in that LAST_TREE is the root instead of FIRST_TREE. (See Figure 7-6).

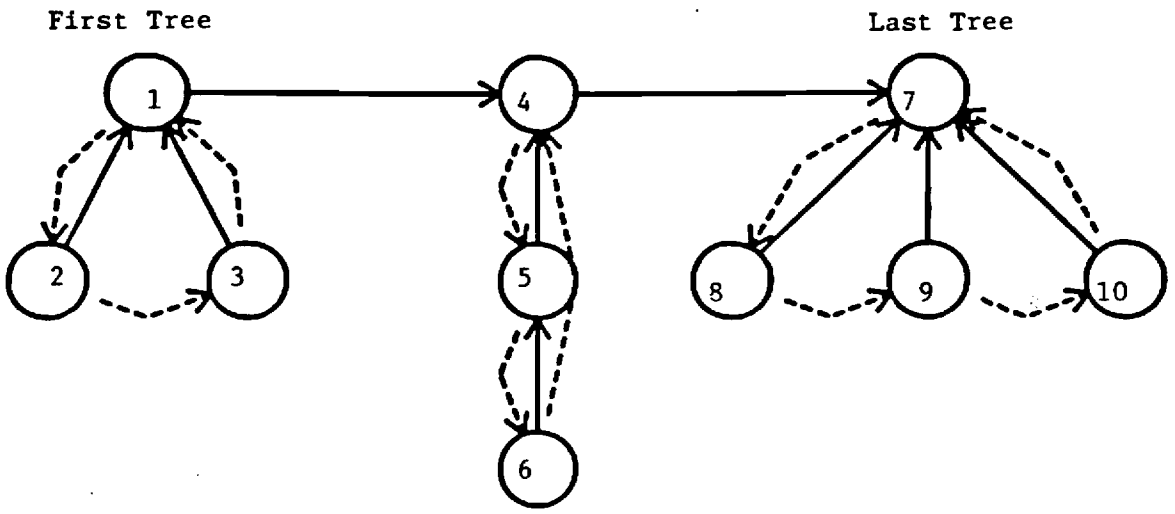
METHOD:

Reverse the PRED values from FIRST_TREE to LAST_TREE;

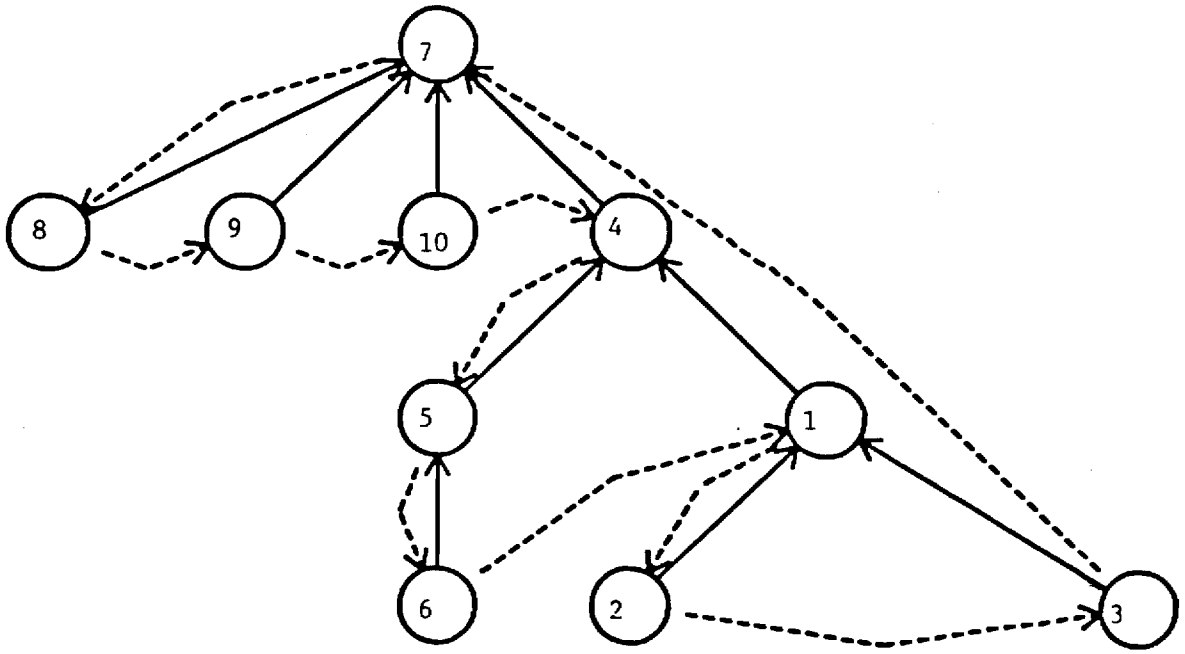
CYC_REROOT(LAST_TREE,FIRST_TREE);

DISCUSSION:

Since CYC_REROOT reverses the PRED values, then by reversing the PRED values before calling CYC_REROOT, the PRED values remain unaffected.



Before



After Cyc-Reroot (1,7)

Figure 7-6. Cyc-Reroot Routine

7.1.3 The Pivot Routines

7.1.3.1 Pivot Type 1

Since the exiting and entering arc are the same, no basis structures other than FLOW must be updated.

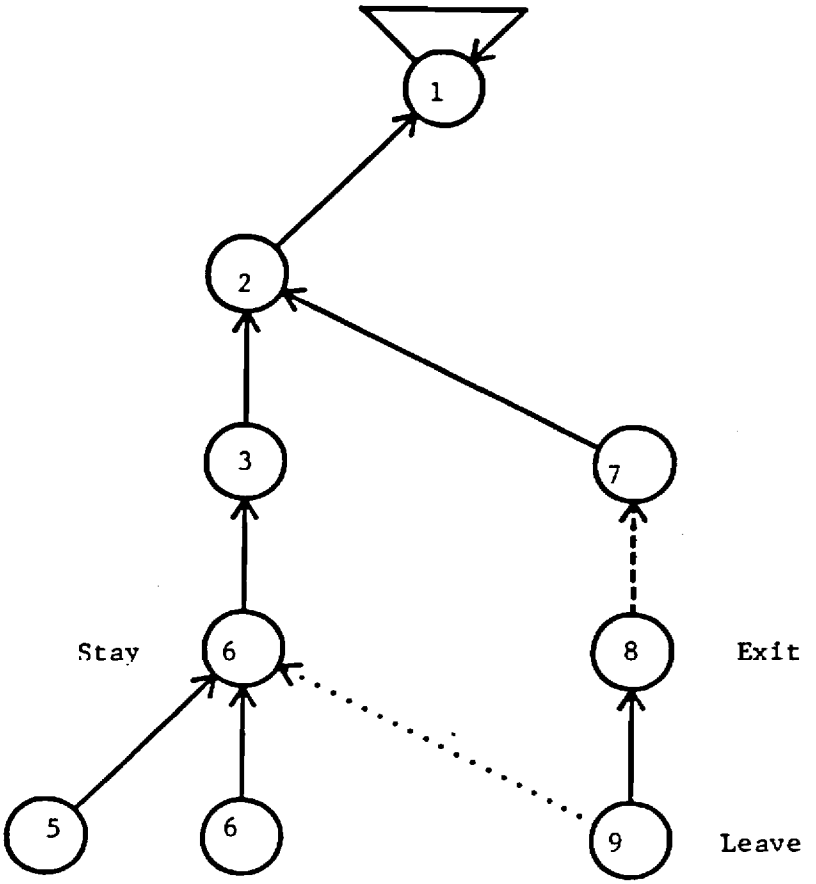
7.1.3.2 Pivot Type 2

Pivot type 2 can occur in one of three cases: entering head and tail are in the same tree, they are in different trees but the same component, or they are in different components. In all cases, the exiting arc is on either the path from the entering tail node to the cycle or the path from the entering head node to the cycle, but not both. The LEAVE node is defined to be the node associated with the path containing the exiting arc. The other node is called the STAY node (see Figure 7-7). EXIT refers to the node that is associated with the exiting arc.

The routine to execute a type 2 pivot is:

```
REROOT(EXIT,LEAVE);
HANG(STAY,LEAVE);
```

See Figure 7-8.



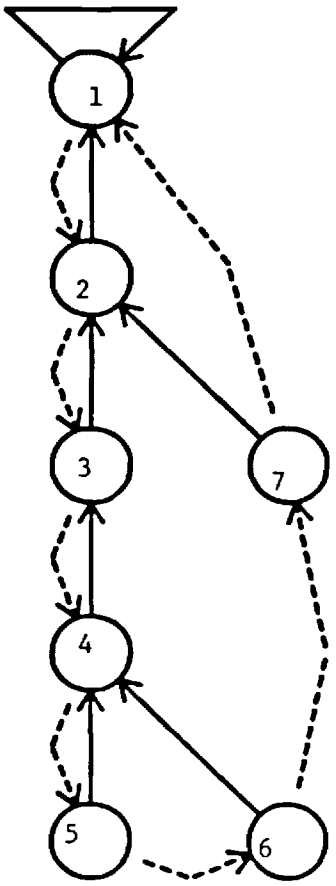
KEY

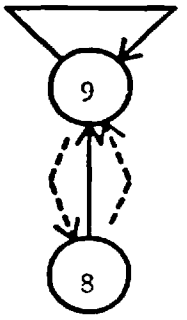
- - - exiting arc

. . . entering arc

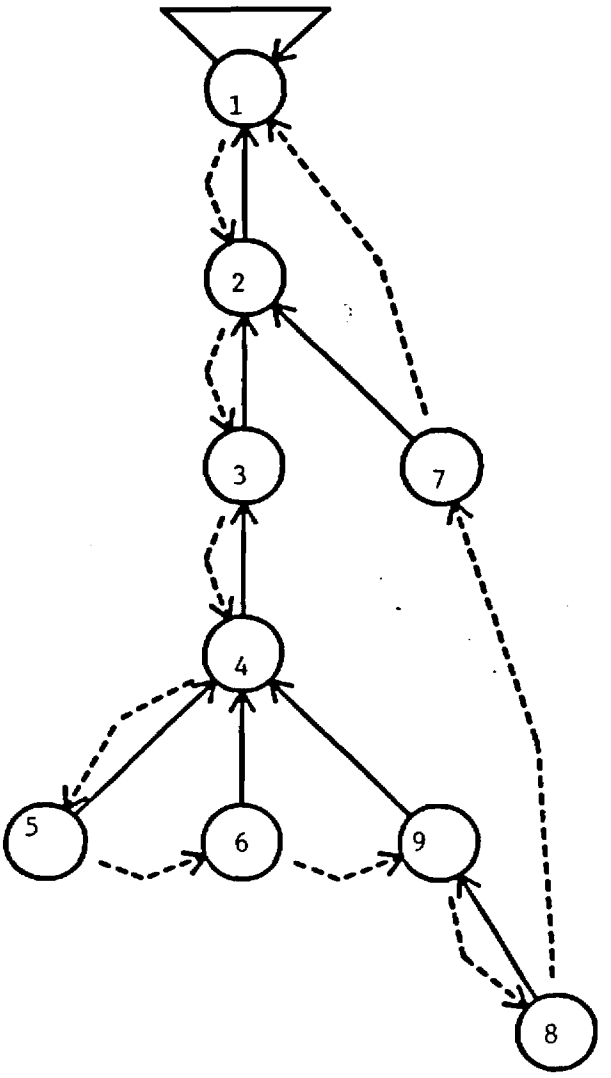
_____ predecessor

Figure 7-7. Pivot Type 2 - Initial Position





After Reroot (8,9)



After Hang (4,9)

Key

Predecessor

Thread

Figure 7-8. Pivot Type 2

7.1.3.3 Pivot Type 3

Pivot type 3 occurs when the entering tail and head are in the same tree and the exiting arc is between MEET and CYC (Figure 7-9). Denote the entering tail and head nodes as ETAIL and EHEAD.

The algorithm for pivot type 3 is:

```
NODE := ETAIL;
PREV := EHEAD;
while NODE <> MEET do
      ISOLATE(NODE);
      TEMP := PRED(NODE);
      PRED(NODE) := PREV;
      PREV := NODE;
      NODE := TEMP;
endwhile;

NODE := EHEAD;
PREV := ETAIL;
while NODE <> MEET do
      ISOLATE(NODE);
      NODE := PRED(NODE);
endwhile;
PRED(MEET) := PREV;

REROOT(EXIT,MEET)
```

See Figure 7-10.

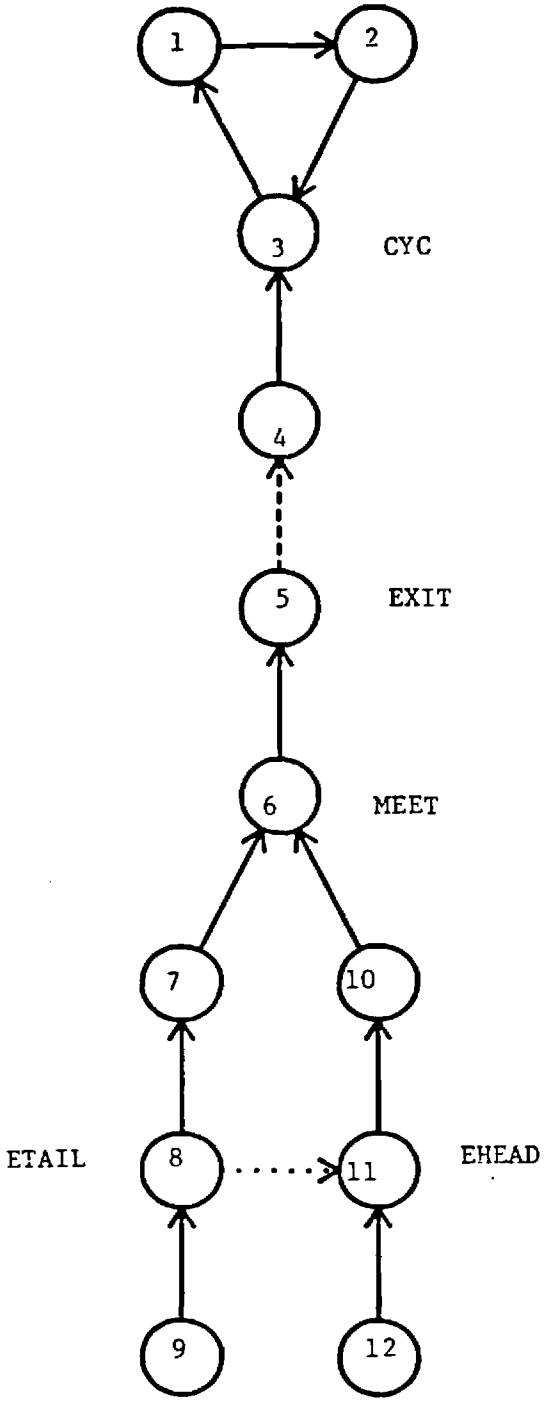


Figure 7-9 Pivot Type 3 - Initial Position

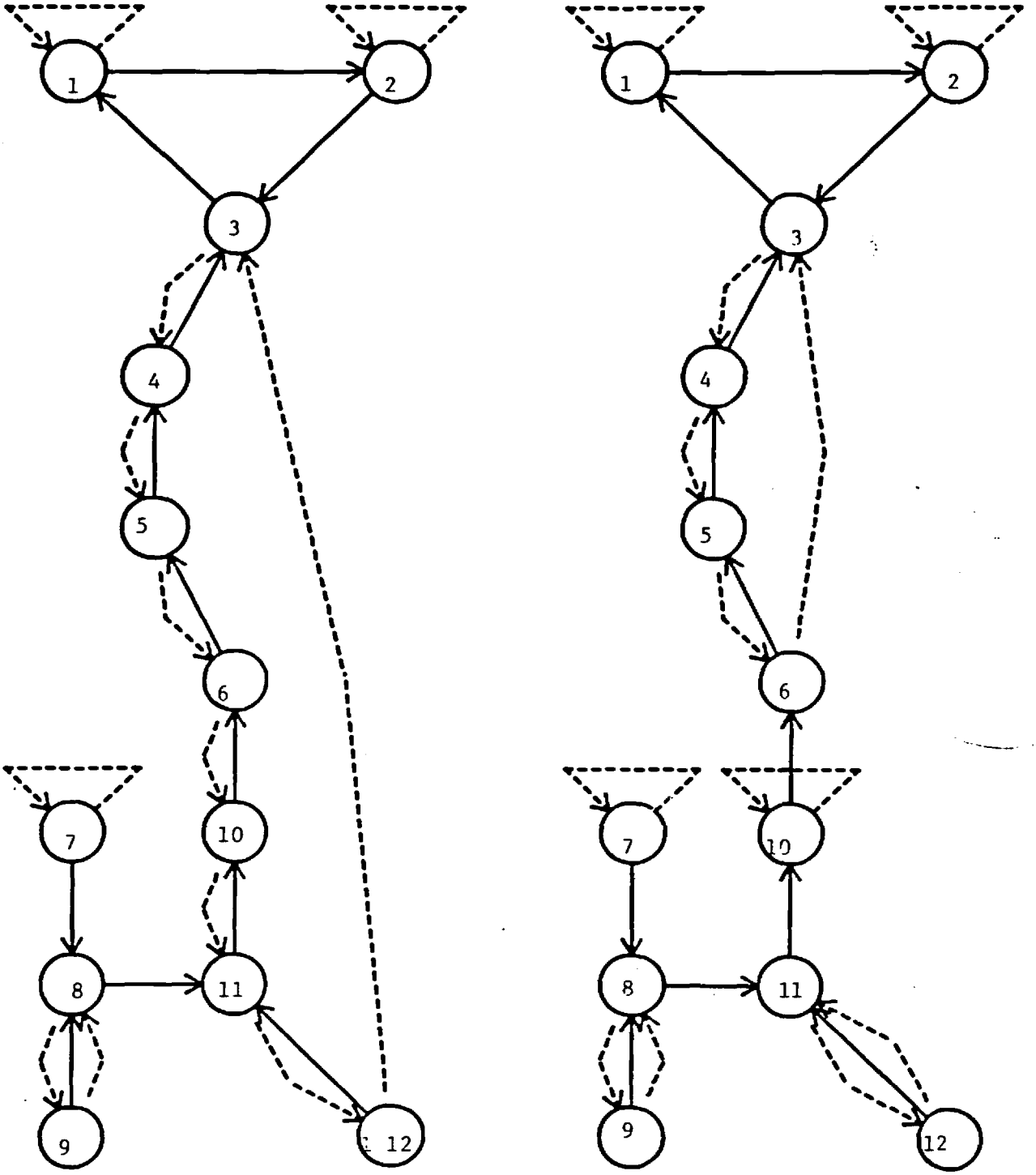
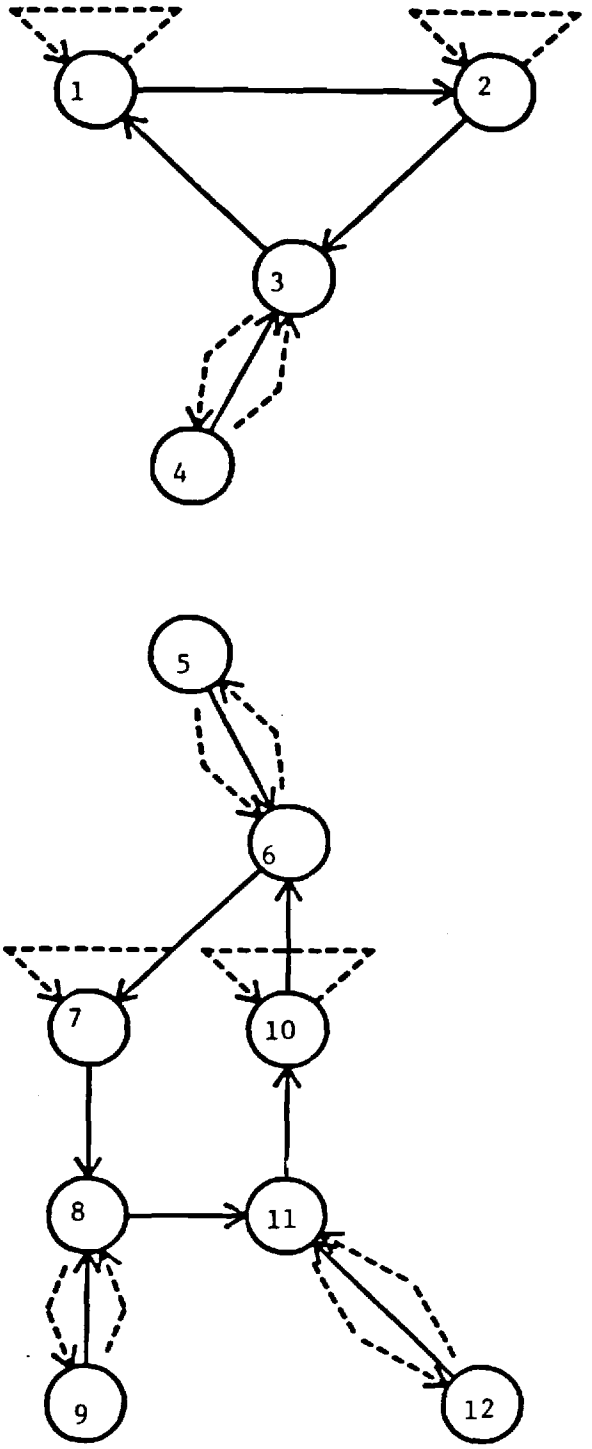


Figure 7-10. Pivot Type 3 During pivot



After Reroot (5,6)

Figure 7-10. Pivot Type 3 During Pivot (Cont.)

7.1.3.4 Pivot Type 4

Pivot type 4 occurs when the entering head and tail nodes occur in the same tree and the exiting arc is on the cycle (Figure 7-11).

The algorithm for pivot type 4 is:

```
NODE := ETAIL;
PREV := EHEAD;
while NODE <> MEET do
      ISOLATE(NODE);
      TEMP := PRED(NODE);
      PRED(NODE) := PREV;
      PREV := NODE;
      NODE := TEMP;
endwhile;

NODE := EHEAD;
PREV := ETAIL;
while NODE <> MEET do
      ISOLATE(NODE);
      PREV := NODE;
      NODE := PRED(NODE);
endwhile;

PRED_EXIT := PRED(EXIT);
REROOT(CYC,MEET);
REV_CYC_REROOT(CYC,EXIT);
CYC_REROOT(PRED_EXIT,CYC);
PRED(MEET) := PREV;
```

See Figure 7-12.

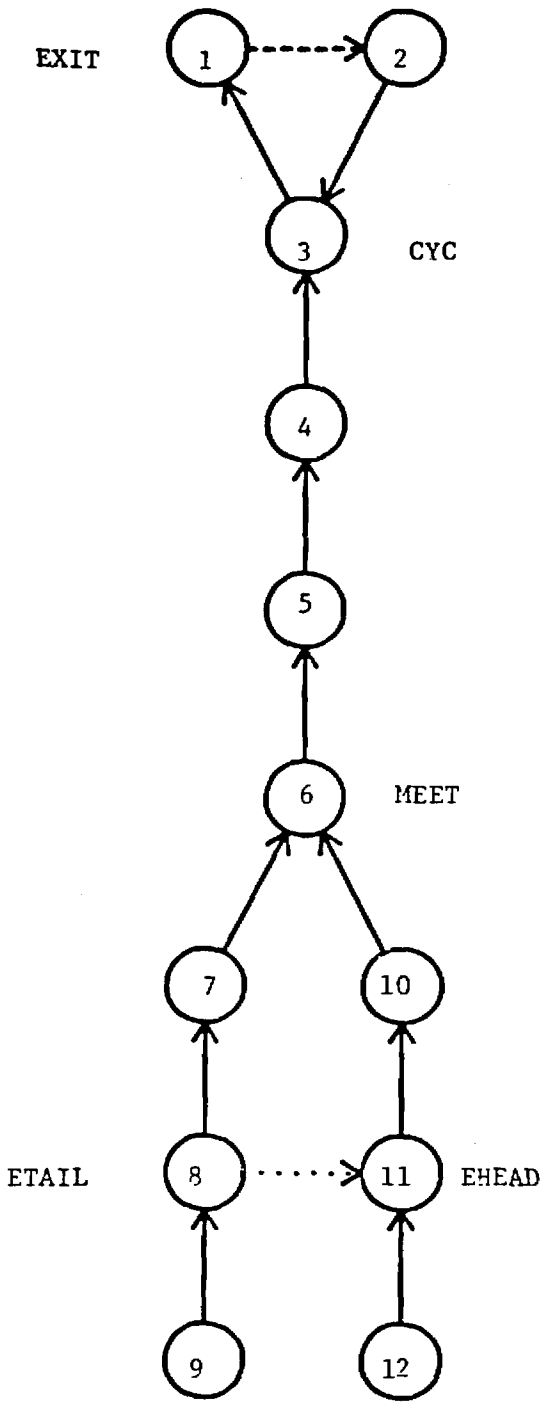


Figure 7-11. Pivot Type 4 - Initial Position

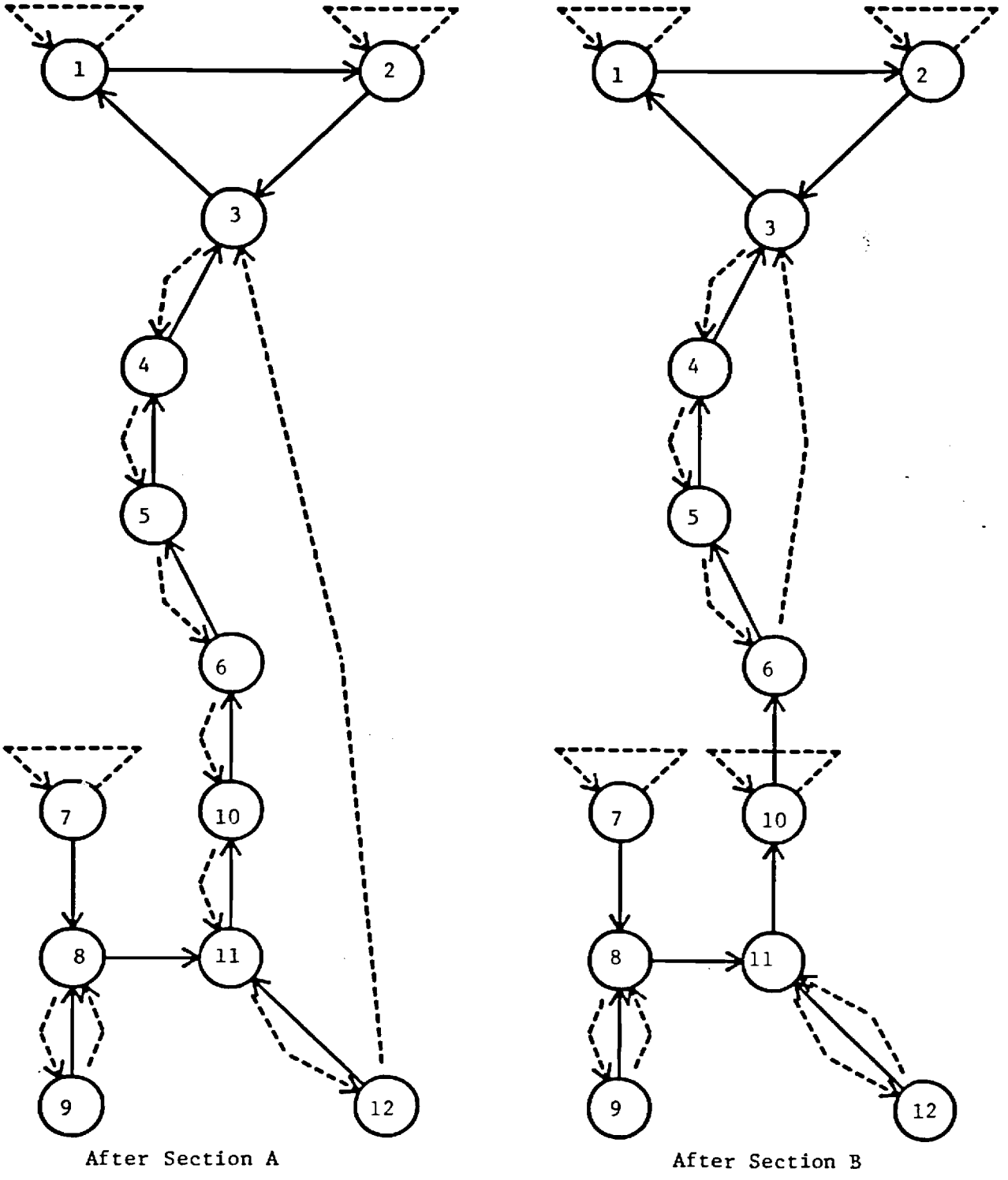
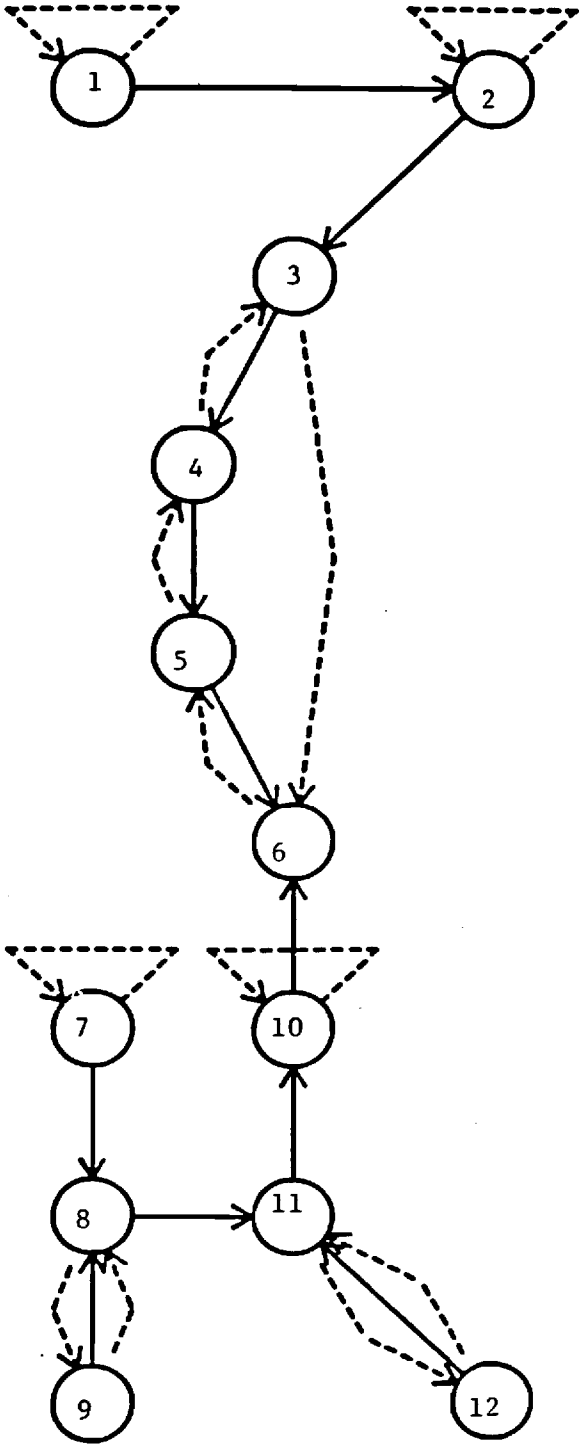
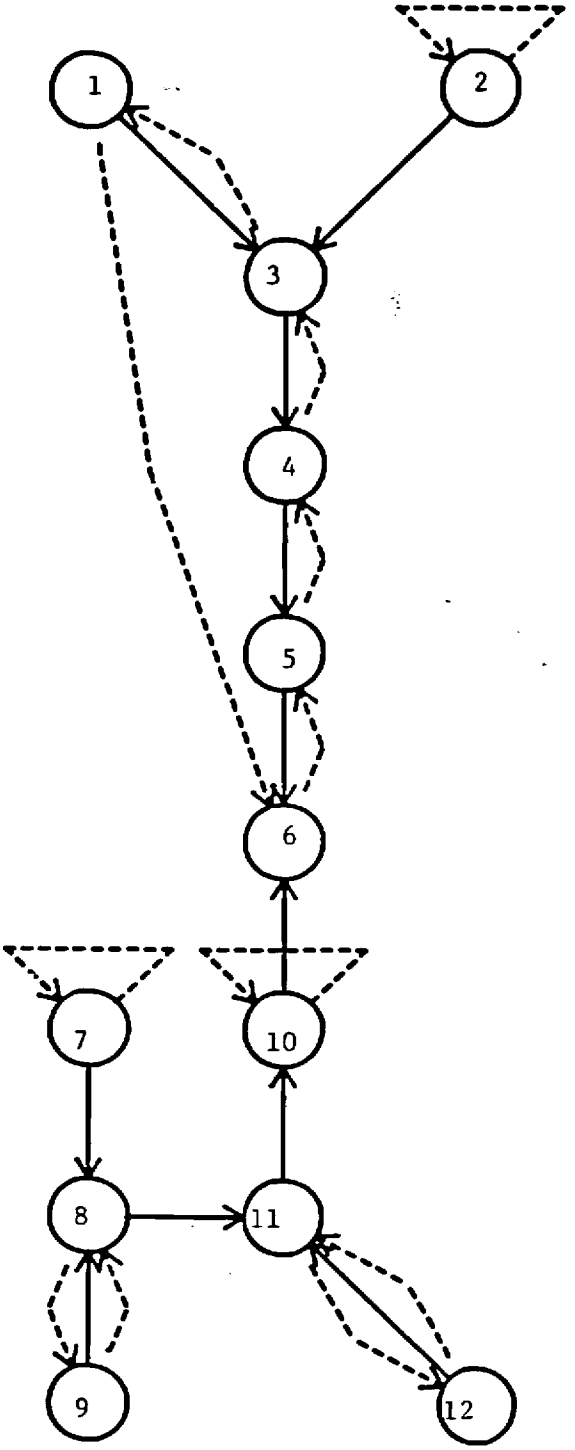


Figure 7-12. Pivot Type 4 - During Pivot

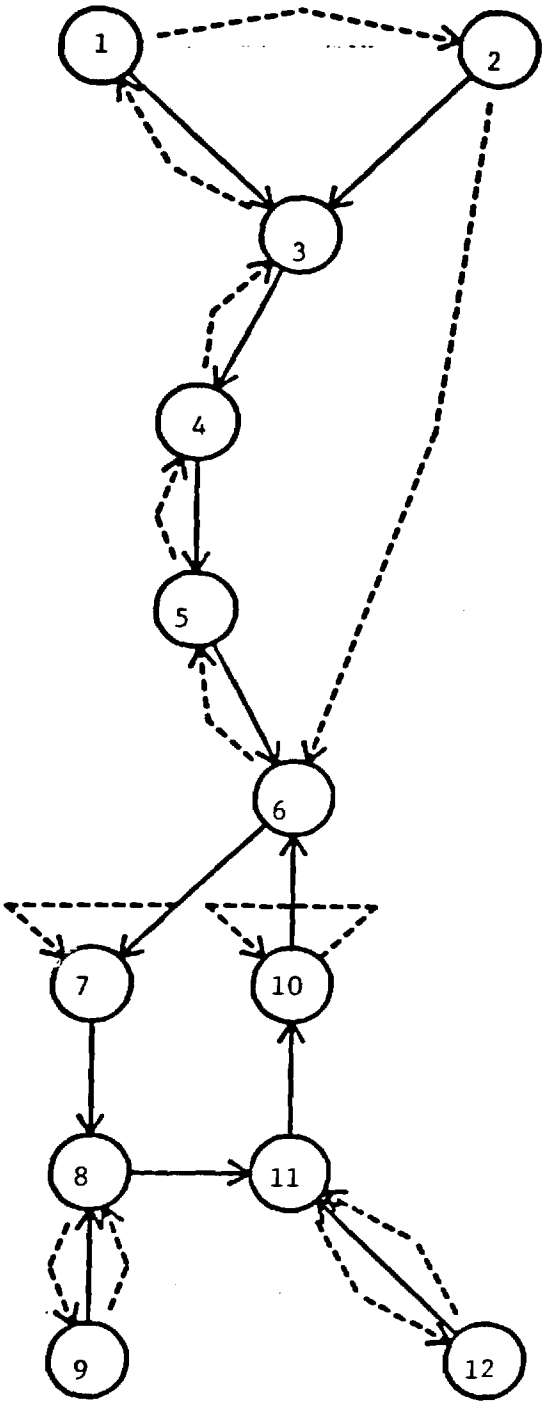


After Reroot (3.6)



After Rev-Cyc-Reroot (3,1)

Figure 7-12. Pivot Type 4 (Cont.)



After Entire Routine

Figure 7-12. Pivot Type 4 (Cont.)

7.1.3.5 Pivot Type 5

Pivot type 5 occurs when the entering head and tail are in the same component, but in different trees. The exiting arc occurs on the cycle (otherwise it would be pivot type 2). If the predecessor path from the entering tail node to (and around) the cycle is compared to the path from the entering head, one of theses path will reach EXIT first. Denote the node with the path to first reach EXIT as LEAVE and the other as STAY. Denote the corresponding cycle nodes as CYC_LEAVE and CYC_STAY (see Figure 7-13).

The algorithm for pivot type 5 is:

```
NODE := LEAVE;
PREV := STAY;
while NODE <> CYC_LEAVE do
     ISOLATE(NODE);
     TEMP := PRED(NODE);
     PRED(NODE) := PREV;
     PREV := NODE;
     NODE := TEMP;
endwhile;

PRED_EXIT := PRED(EXIT);
REV_CYC_REROOT(CYC_LEAVE,EXIT);

NODE := STAY;
PREV := LEAVE;
while NODE <> STAY_CYC do
     ISOLATE(NODE);
     PREV := NODE;
     NODE := PRED(NODE);
endwhile;

CYC_REROOT(PRED_EXIT,CYC);
```

See Figure 7-14.

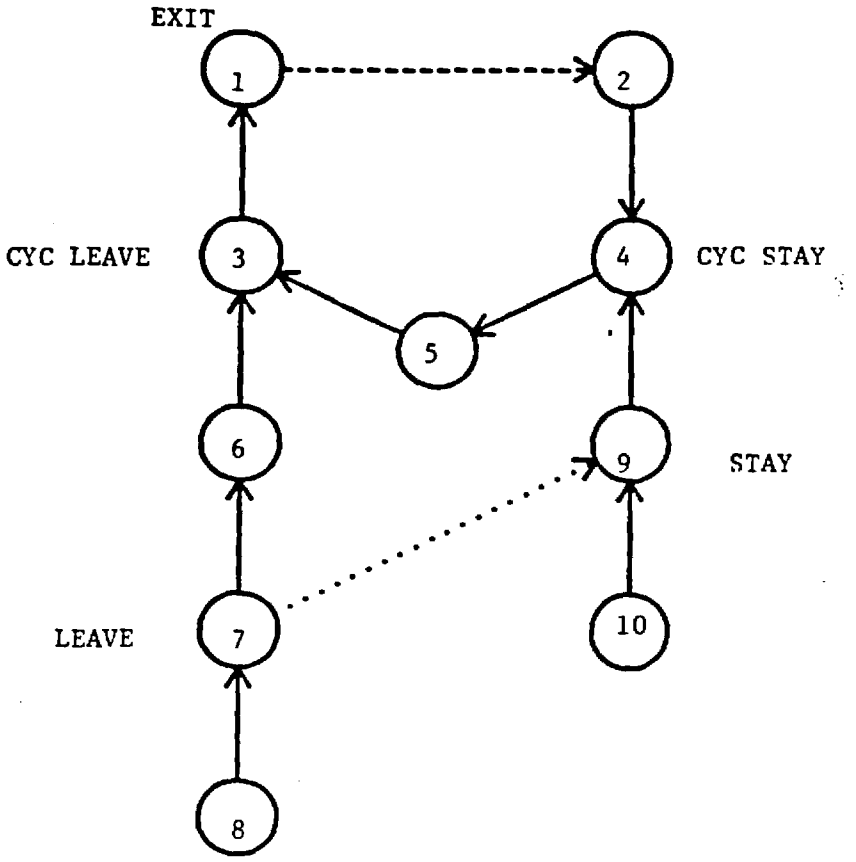
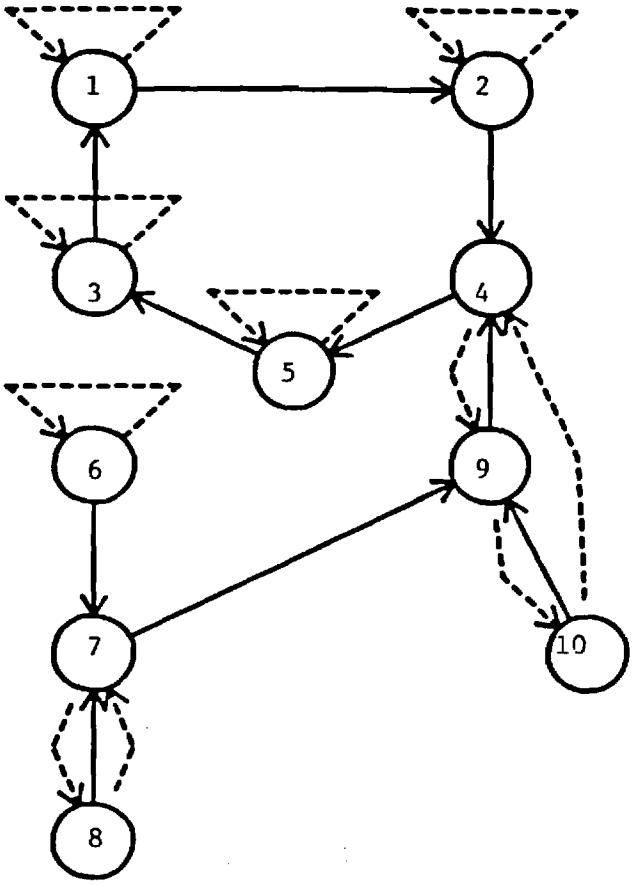
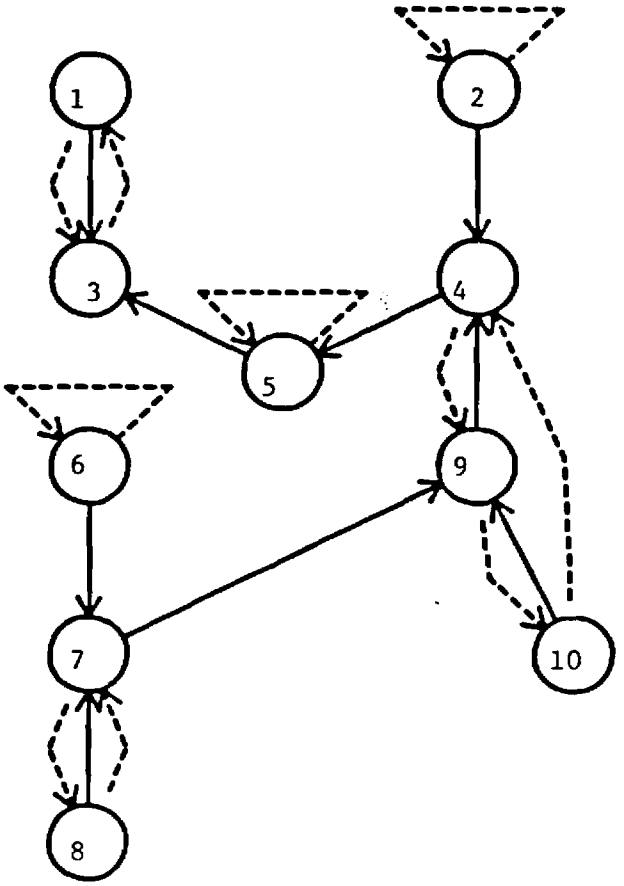


Figure 7-13. Pivot Type 5 - Initial Position

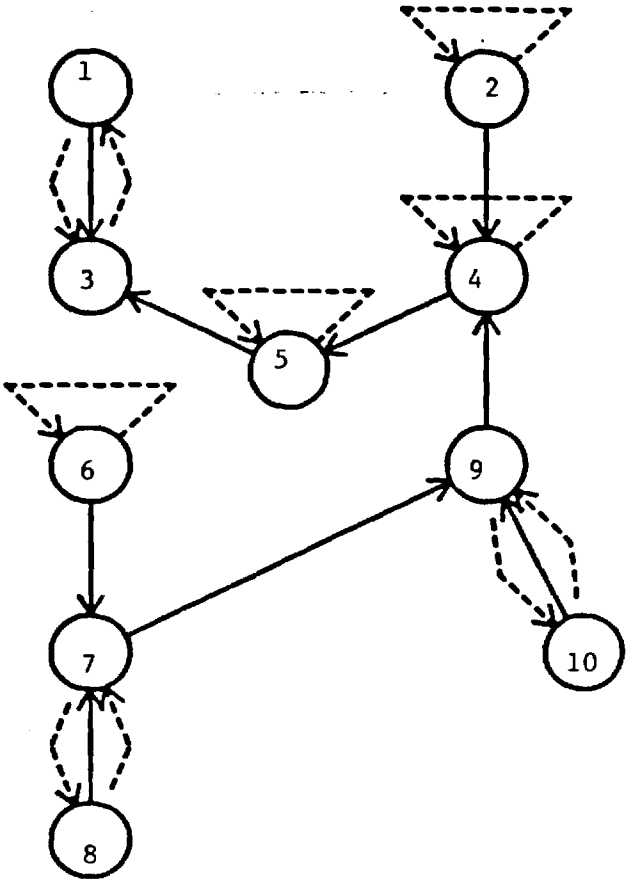


After Section A

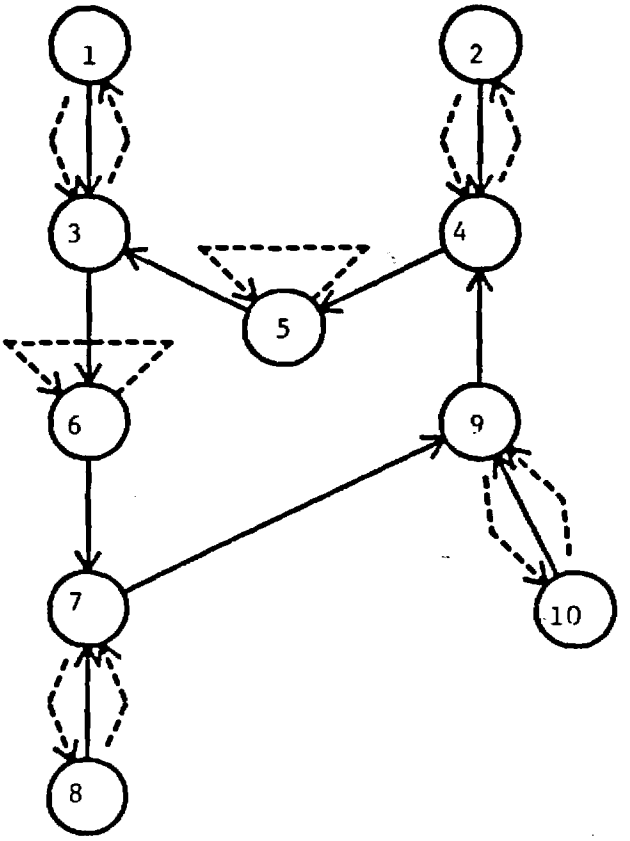


After Rev-Cyc-Reroot (3,1)

Figure 7-14. Pivot Type 5 During Pivot



After Section B



After Pivot

Figure 7-14 Pivot Type 5 During Pivot (cont.)

7.1.3.6 Pivot Type 6

Pivot type 6 occurs when the entering arc has its nodes in different components and the exiting arc occurs on a cycle. (If the exiting arc is not on a cycle then the pivot type is 2).

Denote the node of the entering arc that is in the same component as the exiting arc as LEAVE and the other end as STAY. Let CYC be the cycle node associated with LEAVE. See Figure 7-15.

The algorithm for pivot type 6 is:

```
REROOT(CYC,LEAVE);
PRED_EXIT := PRED(EXIT);
REV_CYC_REROOT(CYC,EXIT);
CYC_REROOT(PRED_EXIT,CYC);
HANG(STAY,LEAVE);
```

See Figure 7-16.

7.2 Updating the Duals

In order to calculate reduced costs quickly, dual values associated with the node are maintained in core. Because the dual variable calculation is a computationally expensive operation, it is fortunate that only a small number of dual values change at each iteration. The duals that change are exactly those whose node receives a new LEVEL value.

The key to calculating duals is that reduced cost for

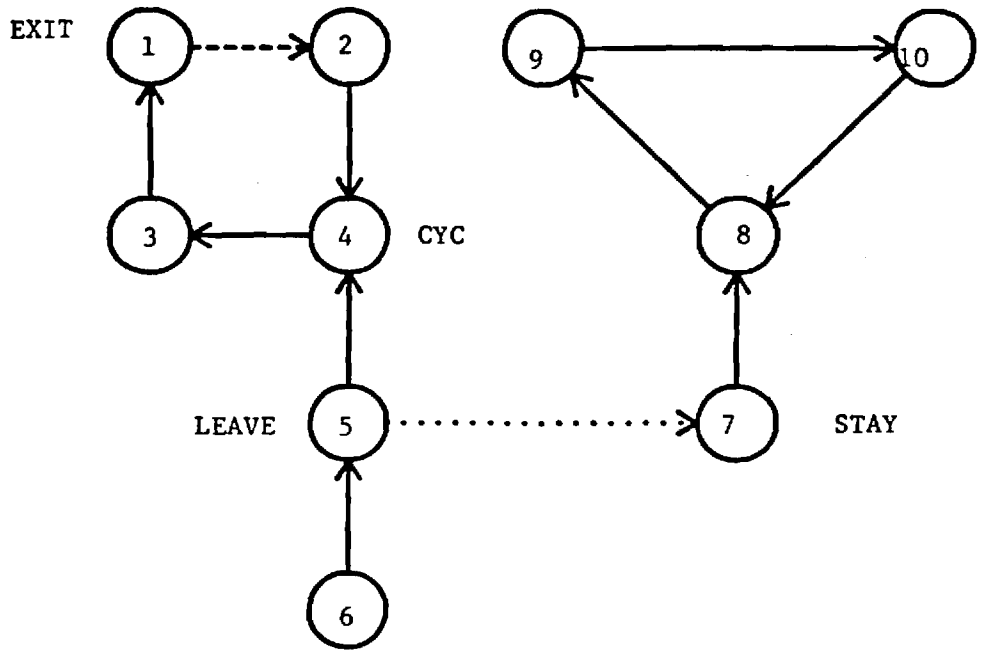
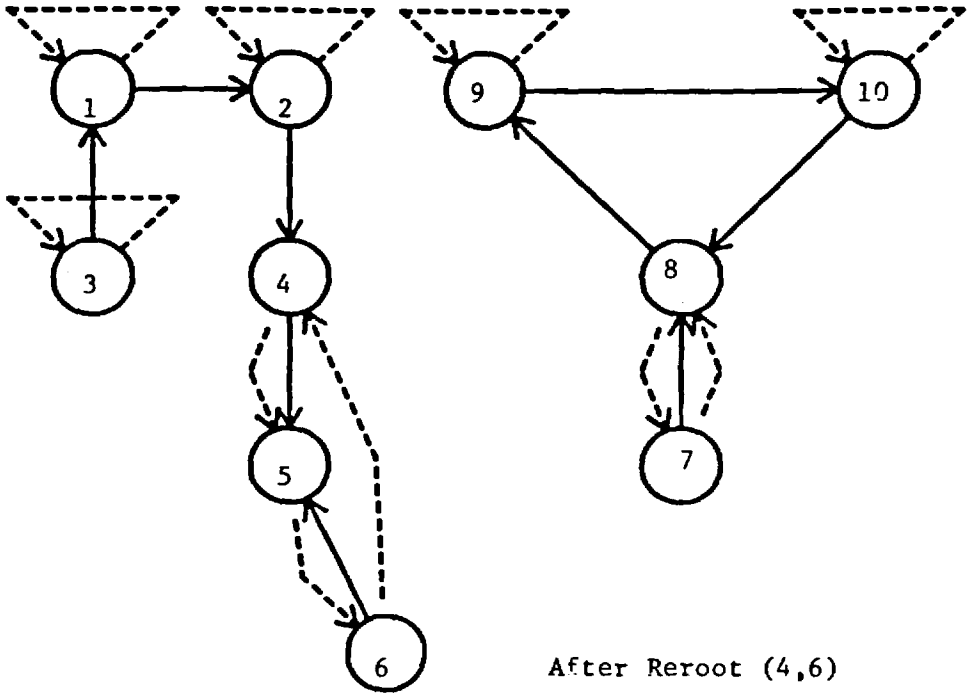
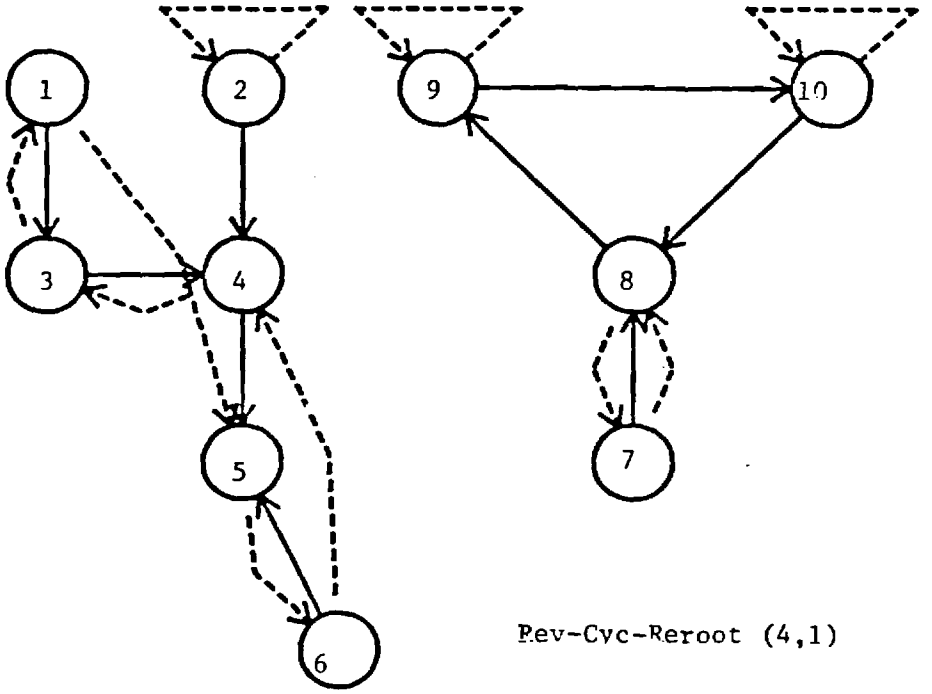


Figure 7-15. Pivot Type 6 - Initial Position

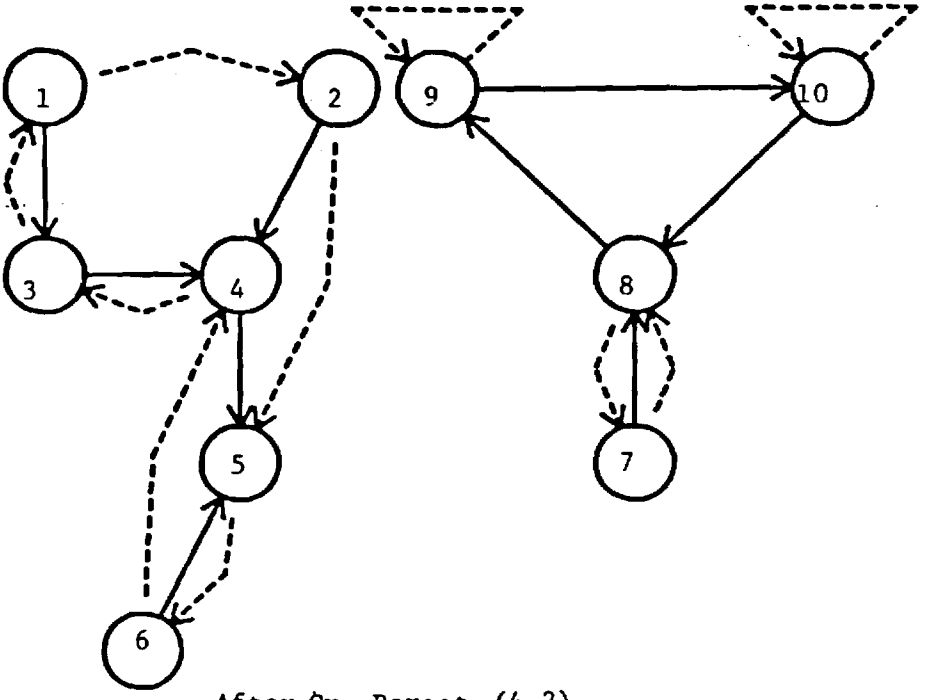


After Reroot (4,6)

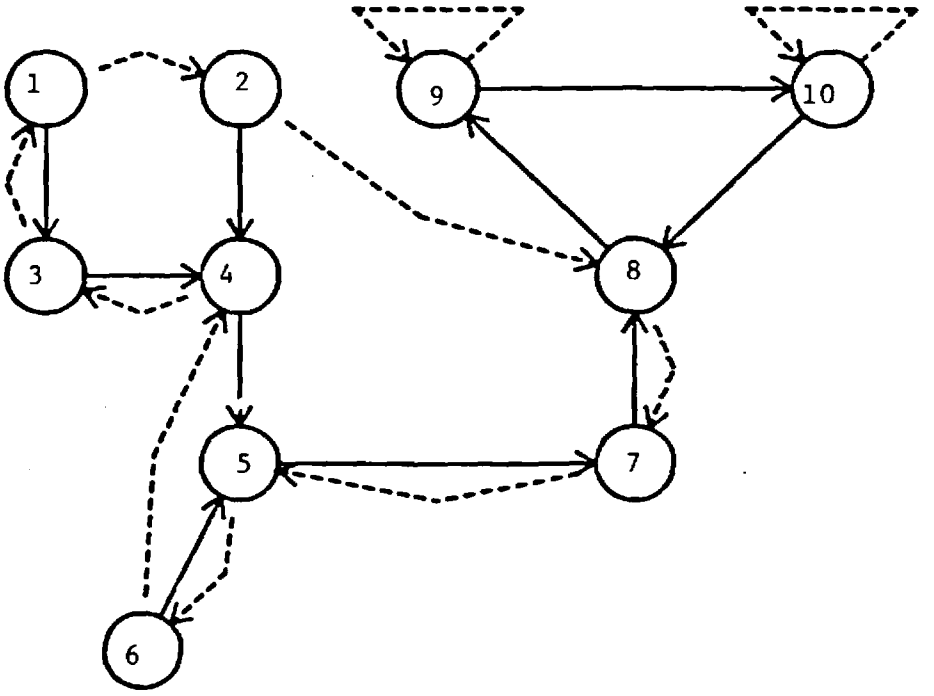


Rev-Cyc-Reroot (4,1)

Figure 7-16. Pivot Type 6 - During Pivot



After Cyc-Reroot (4,2)



After Pivot

Figure 7-16. Pivot Type 6 - During Pivot (Cont.)

arcs in the basis are always zero. The reduced cost calculation is (from Section 5)

MULT * DUAL(HEAD) - DUAL(TAIL) - COST

Duals for generalized networks are uniquely determined. Calculation of duals for nodes on the cycle is involved and is reviewed in Section 7.2.1. The dual for a node off the cycle is based solely on the dual value of its predecessor. This is shown in Section 7.2.2.

7.2.1 Dual Values on the Cycle

Given a cycle with k nodes there are k arcs between them. Each arc creates one reduced cost calculation. Therefore, there are k linear equations to find k unknown duals.

If one dual on the cycle is given, the rest can be obtained by traversing the cycle with the PRED values. The method to find one dual on the cycle follows.

Let the nodes on the cycle to be 1 ... k, with PRED(i) = i+1 and PRED(k) = 1. The following calculates DUAL(1):

```
TOT_SUM := 0;
TOT_MULT := 1;
NODE := 1;
repeat
     if ARC(NODE) < 0 then (* ARC is from PRED(NODE) to NODE *)
          TOT_MULT := TOT_MULT / MULT(NODE);
          TOT_SUM := TOT_SUM - TOT_MULT * COST(NODE);
     else
          TOT_SUM := TOT_SUM + TOT_MULT * COST(NODE);
          TOT_MULT := TOT_MULT * MULT(NODE);
     endif;
     NODE := PRED(NODE);
until (NODE = 1);

DUAL(1) := TOT_SUM / (1-CMULT(1));
```

Duals of cycle nodes associated with self-loops are easier. For this case, the dual is simply the cost divided by the self-loop multiplier.

7.2.2 Dual Values not on the Cycle

The dual value for a node off the cycle is calculated as:

```
if ARC(NODE) < 0 then
     DUAL(NODE) := (DUAL(PRED(NODE)) + COST) / MULT;
else
     DUAL(NODE) := DUAL(PRED(NODE)) * MULT - COST;
endif
```

It is important that the dual variable updates are performed in the correct order. For every node, the dual for the PRED of the node must be calculated before the dual of the node can be calculated. It is for this reason that THREAD is defined to be the preorder traversal. Following the thread will ensure update of the PRED of any node before the node itself; the DUAL update can be accomplished at the same time. This reduces the number of times any node must be examined during a pivot, yielding in a significant reduction in computation time.

7.2 Updating the Flows

The final structure to update is the basic flow values. Most of the work in this update was accomplished during the calculation of the exiting arc. The updated column was the change in flow on the basic arcs if one unit of flow was put on the entering arc. This

alloved a calculation of the maximum flov that could be placed on the entering arc without violating any bounds. This value is multiplied by the updated column and added to the current flovs to produce the updated flovs.

Algorithmically, it is easier to perform this update while the basis is being updated. As each node is visited it is a simple matter to calculate the amount the flov will change and update FLOW accordingly. Some nodes are not visited during the basis update and those nodes must be visited solely to update the flovs. For instance, in pivot type 1, no basis update is performed but the flovs must be updated. All nodes visited in the calculation of the exiting arc must be revisited to update the flovs.

8.0 OTHER CONCERNS

Major portions of the primal simplex method have been covered. Given a basis it is possible to find an arc to enter the basis, determine the arc to exit the basis and update the basis. The only other step of the primal simplex method is to find an initial basis. Section 8.1 gives two methods for doing this.

One other concern is the efficiency of the generalized network primal simplex method when solving a pure network. It is generally thought that a pure network code will execute two or three times faster than a generalized network code. Some of this improvement is due to the fact that generalized networks must keep track of the multipliers while the pure network has one less piece of data for each arc. Some efficiency relates to the structure of the pivots used with pure networks. This pivot and basis structure can be used within a generalized network code, so pure networks and "almost" pure networks can be solved more quickly than generalized networks with the generalized network simplex method. This is detailed in Section 8.2.

8.1 Initial Basis

The starting basis has a large effect on time required by the primal simplex method. Ideally, if the optimal basis is used as the starting basis, then no pivots need be performed. The concept of quality of the starting basis is difficult to quantify, although it seems reasonable to assume that a good starting basis is one which

has a large number of arcs that are also in the optimal basis.

If too much time is spent in trying to find a good starting basis, the total execution time might be larger than using a poorer, but easier to find, basis.

An initial basis must satisfy the following conditions:

1) There must be one arc in the basis for each node.

2) Each component generated by the arcs in the basis must have exactly one cycle, which may be a self-loop.

3) The net flows into and out of each node must be equal to the supply or demand at that node.

4) The flow on any arc not in the basis must be either zero or the capacity of the arc.

5) The flow on any arc must be between zero and the capacity of the arc inclusive.

Section 8.1.1 describes the easiest basis to find: the artificial start basis. By spending essentially no time in creating the initial basis, this basis makes no attempt to guess which arcs will be in the optimal basis.

Section 8.1.2 discusses basis creation methods that try to guess which arcs are optimal. These advanced start methods are often problem specific, although some general purpose methods are possible.

8.1.1 Artificial Start

Associated with each node in a generalized network is a self-loop, representing the slack or surplus variable at the node. If the supply or demand at the node must be satisfied then there is a large cost on the self-loop, otherwise the cost is zero. The artificial

start basis is as follows:

1) The flow on every self-loop is the supply or demand at the node.

2) The flow on every other arc is zero.

The arcs in (1) above form the basis. This basis satisfies the five conditions given in Section 8.1. It is also easy to create. Furthermore the PRED, THREAD, LEVEL and RTHREAD are very easy to calculate. Figure 8-1 illustrates this for the network in Figure 2-3.

The disadvantage of this method is that no attempt is made to determine which arcs will likely occur in the optimal basis. The number of pivots required after an artificial start is probably more than the number required after other types of starts.

8.1.2 Advanced Start

A basis that attempts to contain arcs that will occur in the optimal basis is referred to as an advanced start. Advanced starts can often be determined from the structure of the particular problem types that are being solved. Many, although not all, advanced start strategies adopt the following form:

1) Sort the arcs in decreasing order of attractiveness (likelihood to be in optimal basis)
2) Using the most attractive unexamined arc, place the maximum amount of flow that will keep feasibility at the nodes and the arc on the arc.
3) Repeat Step 2 until all of the arcs have been examined.

Algorithms that adopt the above strategy and place all unallocated supply and demand on the self-loops create a basis that

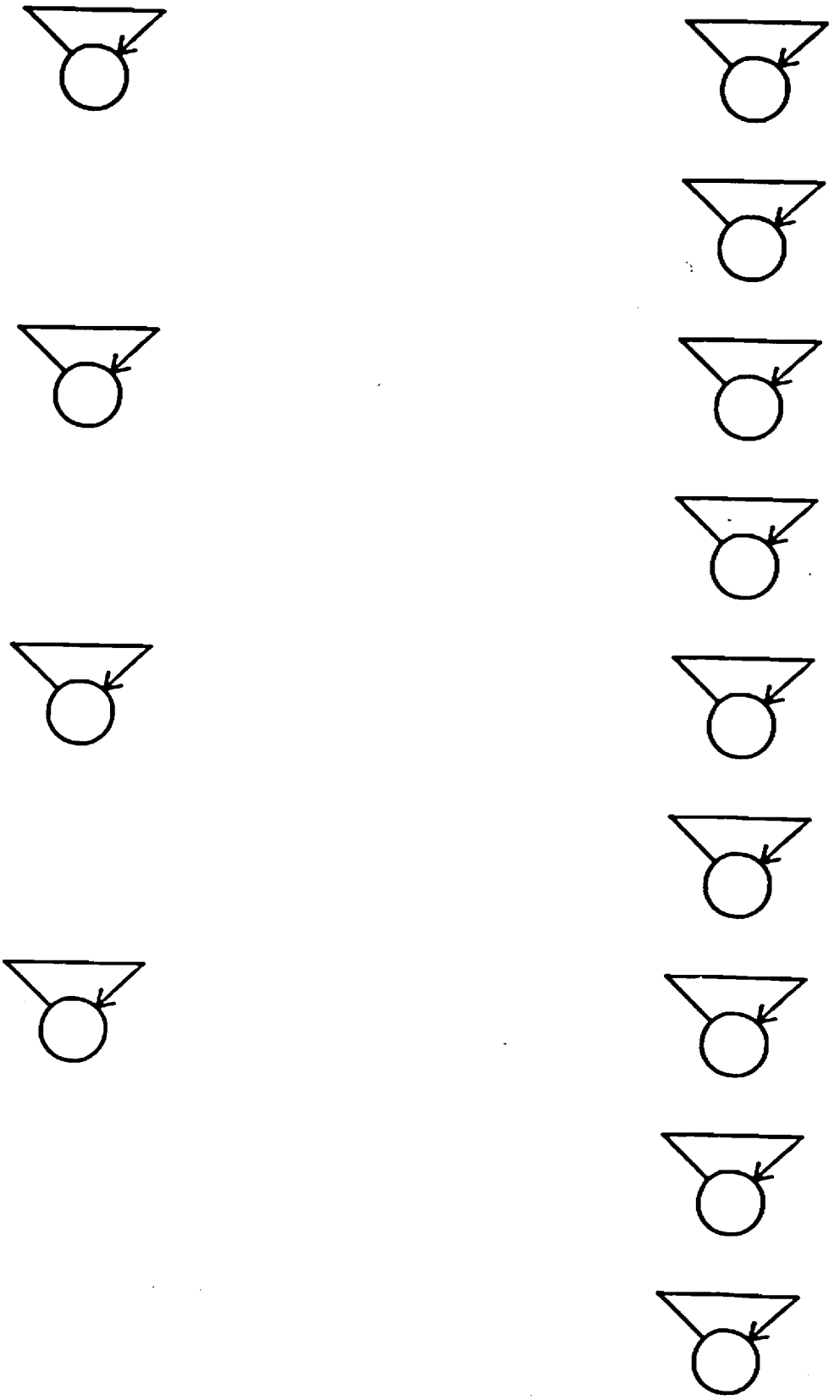


Figure 8-1. Artificial Basis

satisfy the five conditions in Section 8.1 (possibly some self-loops with zero flow will have to be added to the basis). If easy methods of calculating attractiveness are available and if attractiveness is a true measure of the likelihood of being in the optimal basis then advanced starts of this form are generally fairly effective.

In Figure 8-2 the arcs in Figure 2-3 are ordered by increasing cost and an advanced start basis is formed.

8.1.3 Specialization for MRMATE

Arcs of the MRMATE model are already sorted into three classes based on attractiveness. This permits an easy advanced start. Arcs in the zero cost arc file are placed into the basis first. Next, low cost arcs are used. Because there are a large number of high cost arcs, it seems better to ignore them in creating an advanced start and let the rest of the simplex algorithm choose which high cost arcs to use by pivotting them into the basis as needed.

8.2 Effect of Pure Network Structure

Pure networks can be solved more quickly than generalized networks. This is for two reasons:

1) Pure networks have multipliers of one. Hence, rather than using multiplications and divisions, the various simplex calculations can assume the MULT is one.

2) The basis structure of pure networks is simpler than that of generalized networks. Each component of a pure network basis must be rooted at a self-loop.

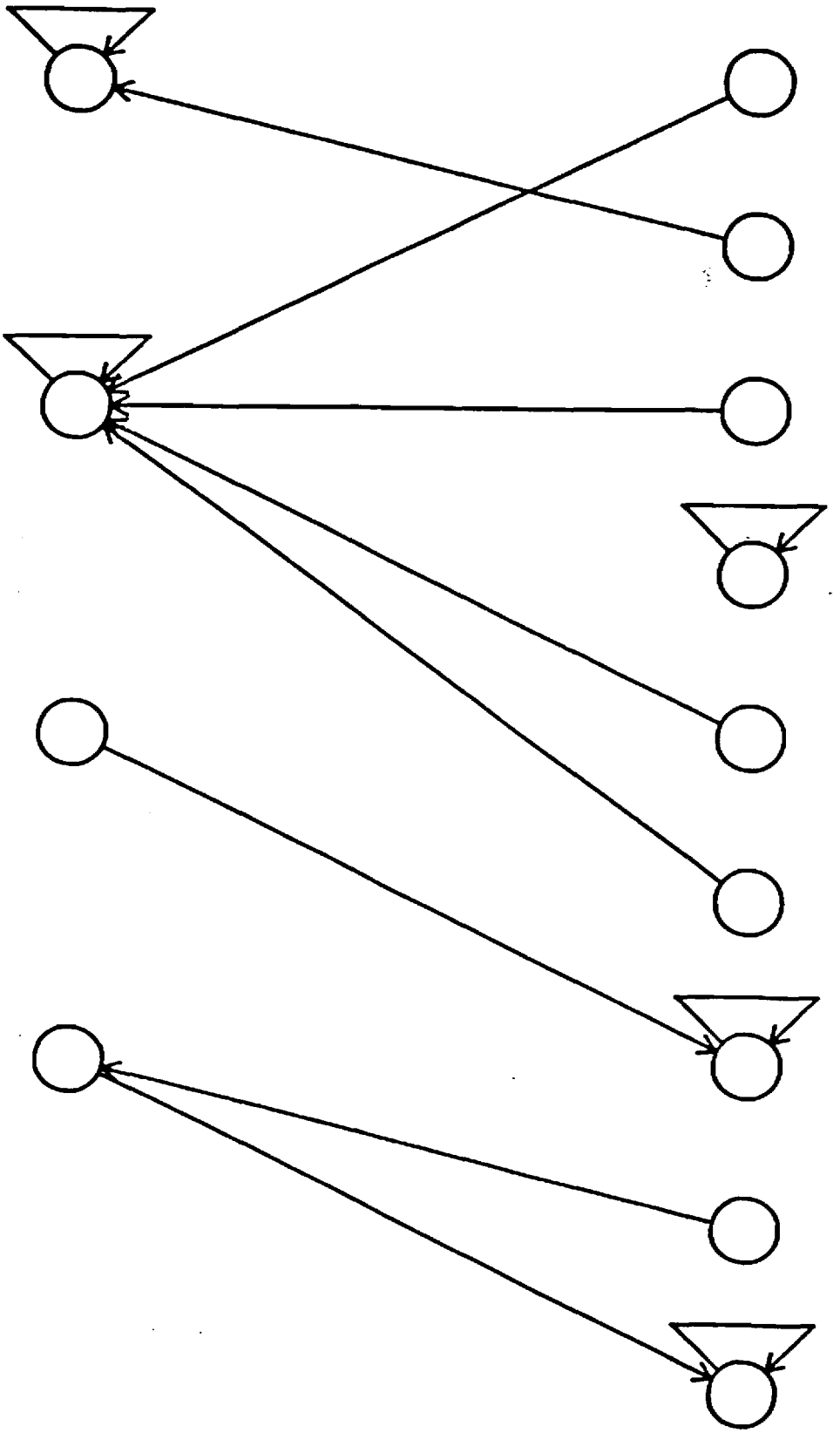


Figure 8-2. Advanced Start Basis

It is possible for a generalized network code to solve pure networks almost as quickly a pure network code by performing the following:

1) A multiplier is compared to 1 before a division or multiplication. If it equals 1 then the division or multiplication is not needed.

2) Data structures are used that are equivalent to pure network data structures in the case of a self-loop root.

Unfortunately, handling the multipliers as suggested in point one will slow down execution for all networks. This decrease in speed may not be excessive but may not be worthwhile if the number of pure arcs is small. Speed increases will occur, however, even in generalized networks, as long as the network has a large proportion of pure arcs.

Data structures used in pure networks are exactly those outlined in Section 4 in the case where the root cycle is a self-loop. Advantages of the basis structure can occur in any generalized network that can be converted to a pure network by scaling the rows and columns of the constraint matrix. The basis advantages can be gained even if the scaling is not performed. An example of a generalized network that can be scaled to a pure network is given in Figure 8-3.

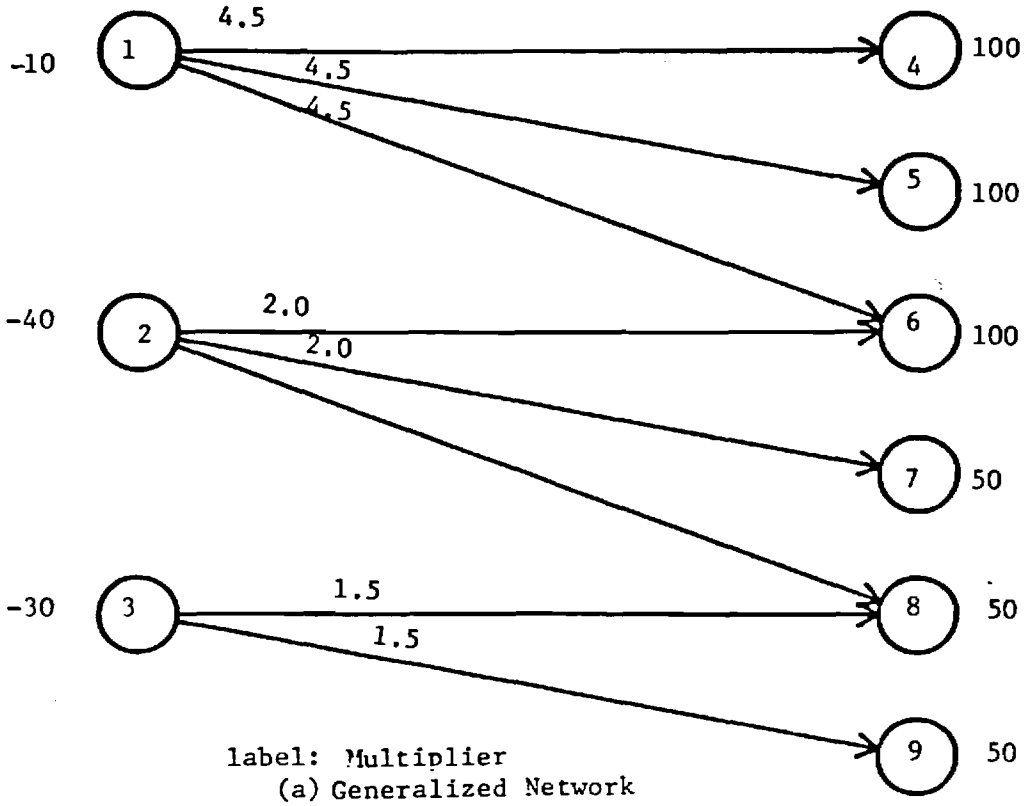
The main advantage of the pure network structure is that all pivots will be of types 1 or 2 (if the self-loops are replaced with artificial arcs and an artificial node, see [3]). No cycles are ever created, so no cycle multipliers need be calculated (Section 4). Calculation of the cycle multiplier is a very expensive operation, since it involves many multiplications and divisions.

It seems likely that a network that is "almost" transformable to a pure network would have many of the same advantages as a network transformable to a pure network. "Almost" must be defined very carefully. For instance, Figure 8-4 shows a network with just one arc with a multiplier not equal to one. However, this network is equivalent to the network in Figure 2-3 and the two networks are solved in almost the same way by a generalized network simplex method. The number of cycle calculations is the same, as is the time required. The reason for this is that non-pure arcs occur in the basis disproportionately for their quantity. This shows the disadvantage of defining "almost" pure networks in terms of the percentage pure arcs.

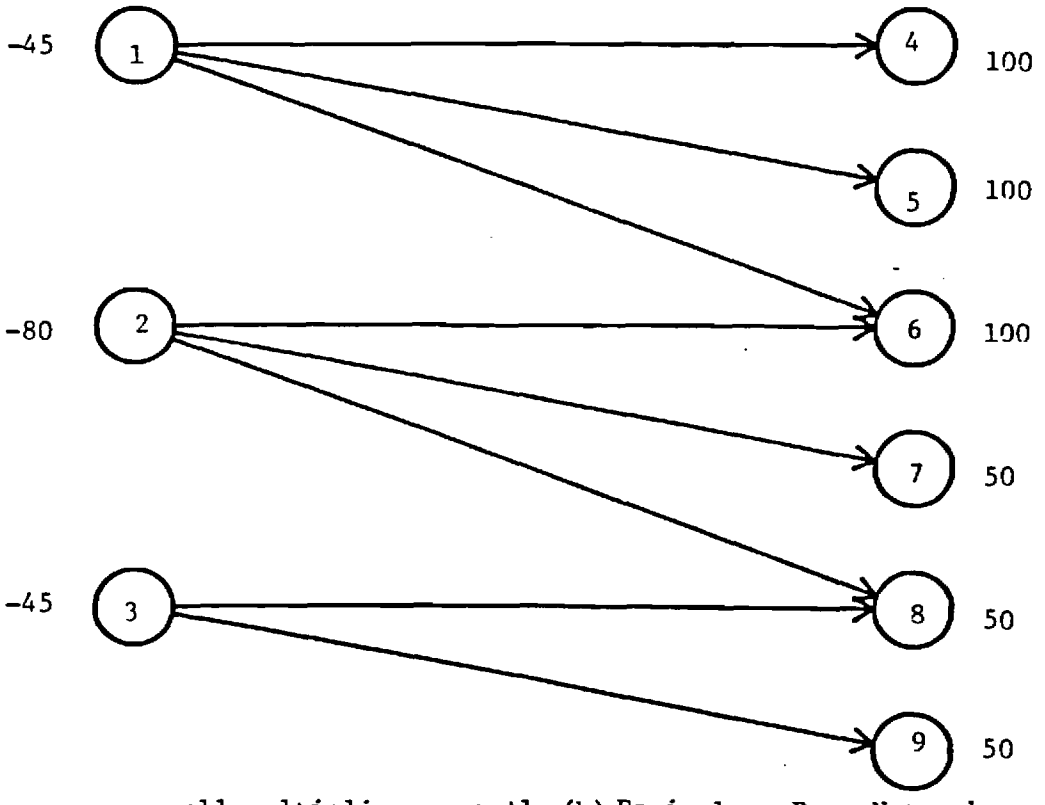
"Almost" pure networks are those whose valid basis tend to have self-loops, rather than cycles. Cycles are formed when the exiting arc occurs on the common part of the backpaths formed by the entering arc. "Almost" pure networks require that for a "typical" basis and entering arc, the exiting arc occurs on the separate parts of the backpath (see Section 6.1).

8.2.1 Specialization for MRMATE

As might be expected, the MRMATE model can be a pure, transformable to pure, or "almost" transformable to pure network. If there are no sea channels then all of the multipliers in the network are one, so a pure network results. If there are no air channels then the network is transformable to a pure network. This is equivalent to using the volume of the movement requirement rather than the weight.



label: Multiplier
(a) Generalized Network



all multipliers are +1 (b) Equivalent Pure Network

Figure 8-3. Generalized Network Transformable to Pure Network

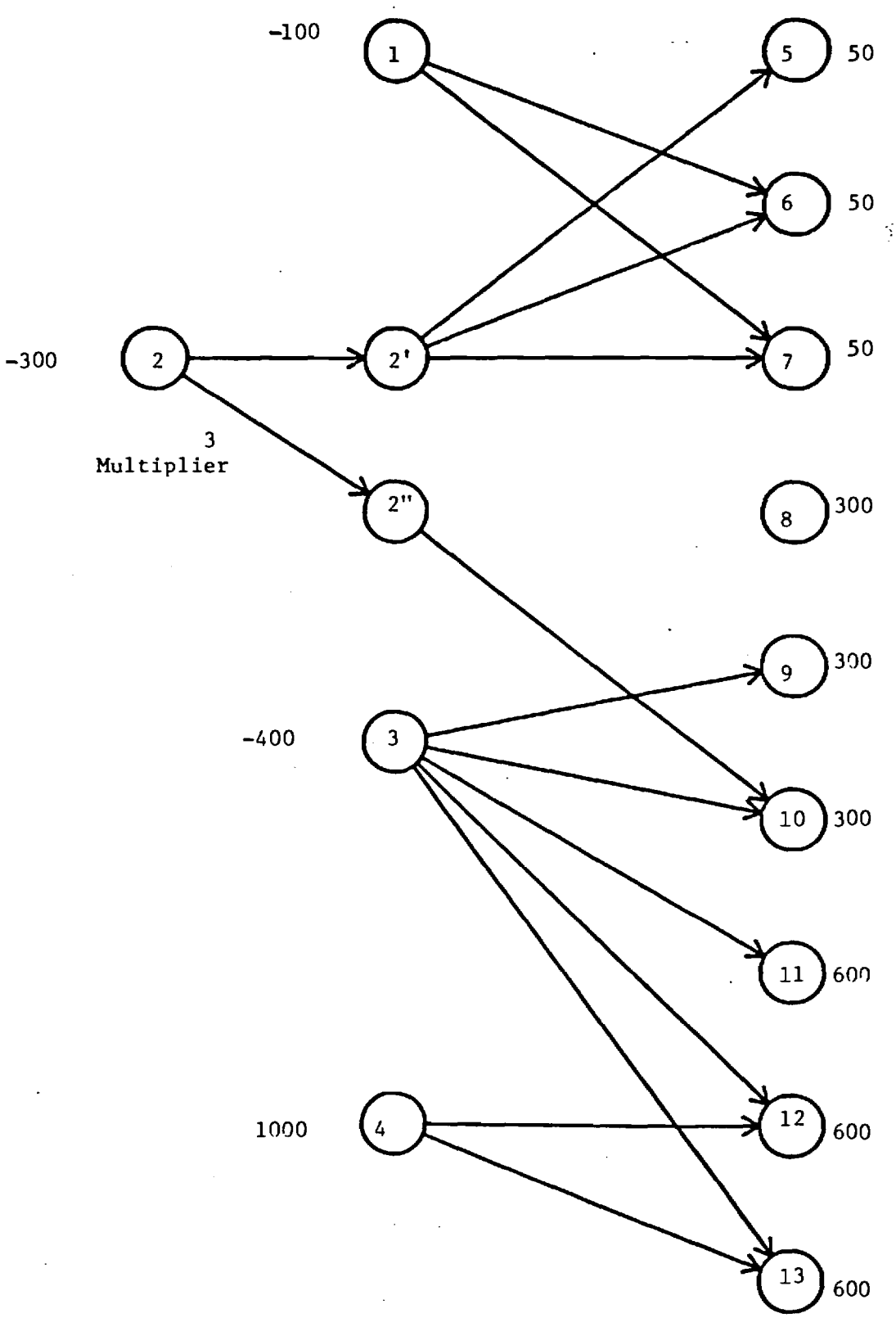


Figure 8-4. Equivalent "Almost Pure" Network

If there are far more air channels than sea channels, or more sea channels than air channels, then the resulting network is "almost" transformable to a pure network as long as the less numerous channels are not disproportionately important. This is due to the following. Given a network with far more air channels than sea channels, a basis and an entering arc, if the channels are equally important, the typical entering arc will be an air channel arc. The backpaths represent the channels that will have to change flows in order to permit the entering arc to join the basis. Typically, this will only involve air channels. This means that the exiting arc cannot occur on the common part of the backpath, since a cycle with a cycle multiplier of 1 would be created. A similar argument holds for more sea channels than air.

How many more is "far more"? Certainly 75 air channels and 25 sea channels will not exhibit much pure network structure, since most backpaths will contain both air and sea channels. Also, 99 sea channels and 1 air channel will exhibit a lot of pure network structure, since most pivots will reassign flow among the sea channels. The effect of pure network structure is further examined in [4].

REFERENCES

[1] Bazaraa, M.S. and J.J. Jarvis, Linear Programming and Network Flows, John Wiley and Sons, New York, 1977.

[2] Brown, G.G. and R. McBride,"Solving Generalized Networks", Management Science, 30, 12, pp 1497-1523.

[3] Jarvis, J.J., H.D. Ratliff, D.E. Eisenstein, A.V. Iyer, W.G. Nulty, and M.A. Trick, "System Description: SYSTEM FOR CLOSURE OPTIMIZATION AND PLANNING EVALUATION (SCOPE)", PDRC Report 84-09, Georgia Institute of Technology, 1985.

[4] Jarvis, J.J., H.D. Ratliff and M.A. Trick, "Generalized Network Results on a Microcomputer", PDRC Report (to be published), Georgia Institute of Technology, 1986.

[5] Kennington, J.L. and R.V. Helgason, Algorithms for Network Programming, John Wiley and Sons, New York, 1980.

Georgia Institute

of

Technology

SEARCH

SCHOOL OF INDUSTRIAL
AND SYSTEMS
ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM
OF GEORGIA
ATLANTA, GEORGIA 30332

Report for:
Joint Deployment Agency
MacDill Air Force Base, FL 33608

Functional Aggregation

and MRMATE formulation

Ananth. V. Iyer
John J. Jarvis
H. Donald Ratliff

PDRC 86-04

Report by:
School of Industrial and Systems Engineering
Georgia Institute Of Technology
Atlanta, GA 30332

This work is supported by the Office of Naval Research under Contract No. N00014-85-C-0797 and N00014-83-K-0147. Reproduction in whole or part is permitted for any purpose of the U. S. Government.

CONTENTS

List of Figures

1.0 INTRODUCTION

The SCOPE (and MODES) system consists of the LIFTCAP and MRMATE problems linked via sensitivity information. PDRC report 85-06 discussed a re-formulation of MRMATE as an approach to generate satisfactory solutions to the MRMATE problem at each iteration.

This report outlines application of these ideas to the solution of MRMATE. Specifically, the effect of aggregation across time, channels, and MRs is analyzed. This report includes a description of the network structure of MRMATE, the significance of MRMATE nodes, arcs and the limits and costs on them. It also outlines various reasons for the aggregation of MRMATE.

An example MRMATE problem, provides a better understanding of the type of data required in the setup of MRMATE and the types of logical aggregation parameters that could be considered. Effects of the aggregation parameter on problem formulation are discussed and illustrated for the example problem considered.

2.0 MRMATE STRUCTURE AND THE NEED FOR AGGREGATION

2.1 MRMATE and its role in SCOPE (MODES):

The MODES deployment system consists of the two problems LIFTCAP and MRMATE which interact to create channel configurations and movement requirement transportation plans for a deployment scenario. Descriptions of LIFTCAP and MRMATE and their details are provided in Chapter 4 of PDRC Report 84-09. This section provides a brief description of MRMATE in the development of a deployment scenario.

MRMATE accepts a set of channels configured by LIFTCAP and time expands them according to the planning increments established by the modeler. It then creates a network model from information regarding cargo categories, channels for a cargo type, the time window for a MR at its destination, and the priority ranking of MRs. The MRMATE model then allocates MRs to channels so that as many of the MRs as possible arrive at their destinations within desired time windows.

The solution to MRMATE is fed back to LIFTCAP which, in turn, re-configures channels in order to move towards global optimality of the overall solution to the deployment scenario.

2.2 Network structure of MRMATE:

The MRMATE model is created after LIFTCAP provides a channel configuration. MRMATE accepts movement requirement data regarding cargo type, quantity, point of origin, destination, and time window at the destination. It also receives channel types and their capabilities. Each channel represents a particular mode and route from a POE (point of embarkation) to a POD (point

of disembarkation).

The MRMATE model is a transportation problem with'source' nodes representing movement requirements; the 'supply' represents the quantity to be transported.'sink' nodes represent channel capability over time. Each LIFTCAP derived channel is represented by T nodes in MRMATE, where T represents the number of planning periods being modeled. The capacity of each MRMATE channel node is the capability generated by LIFTCAP factored over the number of days the MRMATE channel node represents. The transportation problem 'arcs' represent feasible allocations of movement requirements to channels. A generic model of cargo and asset type allocation restrictions is shown in Figure 1. In this example the restrictions mean that there are arcs only between outsized cargo and outsized channels; oversized cargo and outsized and oversized channels; and bulk cargo and outsized, oversized, and bulk channels.

The objective in this example is to deliver the movement requirement requirements to their required destinations within the time windows specified. This objective is modeled by a (convex) cost function which places a cost on a MR when it is allocated to a channel outside the MR's time window. The cost increases with the distance from the window. A low cost (often zero) is applied to movements scheduled within the window. The objective is to attempt to 'push' MRs towards the center of the time window.

2.3 Motivation for aggregation in MRMATE

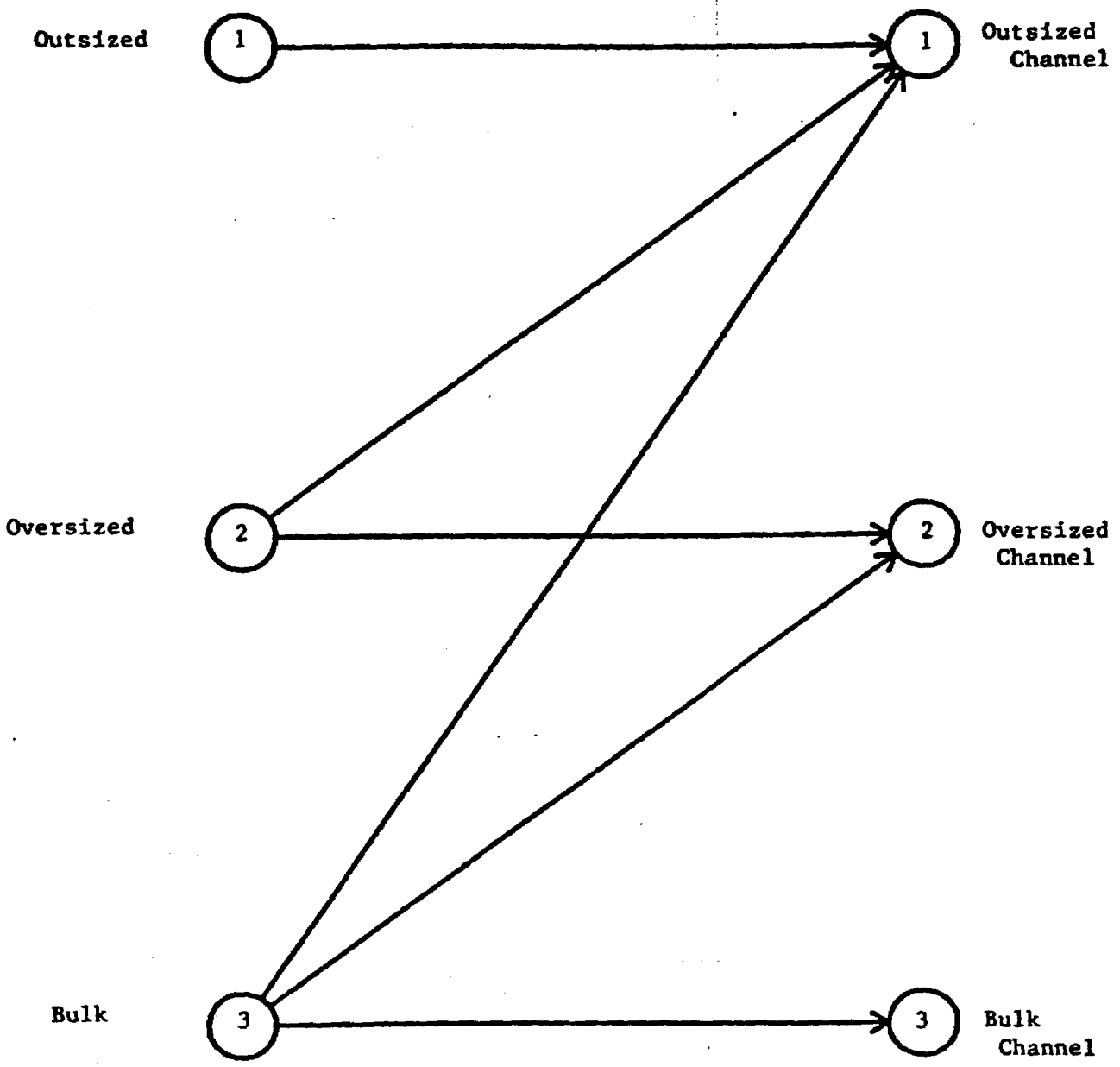


Figure 1. Cargo & Channel Types

2.3.1 Problem Size

Consider the following issue;

Let T = number of time periods,

R = number of movement requirements,

A = number of assets,

I = number of POEs, and

J = number of PODs.

With this notation, MRMATE potentially has R sources, A * I * J * T sink nodes and R * A * I * J * T arcs. For a problem with A = 10 assets, I = 50 POEs, J = 50 PODs, R = 1000 MRs and T = 30 time periods, MRMATE would have 1000 sources, 0.75 million sinks and potentially 750 million arc variables.

The time required for a solution procedure, based on this model, would preclude its usage in a crisis action deployment situation, particularly since it must be re-solved at each iteration of the SCOPE procedure.

2.3.2 Co-ordination

POEs occur at different zones in the U.S. and are therefore are controlled by different regional transportation controllers. If, at each stage, a deployment plan that is 'acceptable' is desired, then inputs from each of the regions regarding the shipments from that region are necessary. In this case, the MRMATE problem must be solved jointly by the different regional controllers, in co-ordination with the Supporting Commander.

2.3.3 Data Quality

In general, the data available is usually subject to some

degree of uncertainty. Since aggregated data tends to be far more stable than the detailed data values, a deterministic model at an aggregate level is more realistic than a detailed one for uncertain data. Once the aggregate model is solved, disaggregation models could use a human interface to analyze the problem to any required degree of satisfaction.

2.3.4 Modeling Issues

There are usually non-quantifiable constraints and objectives which are not included in the model. An example might be unit integrity. After the problem is solved, the aggregate/disaggregate model enables examination of alternate optimal solutions which might be more satisfactory when these additional constraints are included.

These issues suggest the use of a multi-level procedure which decides on a global MR allocation to zones. This problem could utilize the detailed plan generated by the zonal controllers to modify the global allocation, thereby moving the solution towards a satisfactory objective.

3.0 AGGREGATION DIFFICULTIES IN MRMATE

3.1 Aggregation Approaches

Solution techniques for large scale problems emphasize approaches which work with only a portion of the problem at a time. These methods are not particularly affected by increase in the problem size, except for an overall increase in the time required for solution. Aggregation as an approach for the solution of large scale transportation problems was first examined by Balas [1]. He suggested setting up of an aggregate transportation problem by combining similar source and sink nodes to form aggregate nodes. Procedures for decisions regarding which nodes to combine were left to specifics of the problem under consideration. The aggregate solution was disaggregated to yield a solution to the original problem. The iterative step was based on the dual infeasible arcs in the disaggregated problem. A detailed network example in Figure 2 along with its nodes to be aggregated yields the aggregated problem in Figure 3.

Lee[3] examined the case of a minimum cost flow on a general network and extended Balas's ideas to the general network case. Zipkin[5] considers generation of bounds on the 'loss of information' due to aggregation. He shows how the choice of the components in an aggregate cluster could affect the quality of the bounds generated for a general linear programming problem. He also examines the use of aggregation as a tool to setup equivalent formulations of the original linear programming problem, and to initiate the gradual introduction of detail into the problem by iteratively changing weights used in the

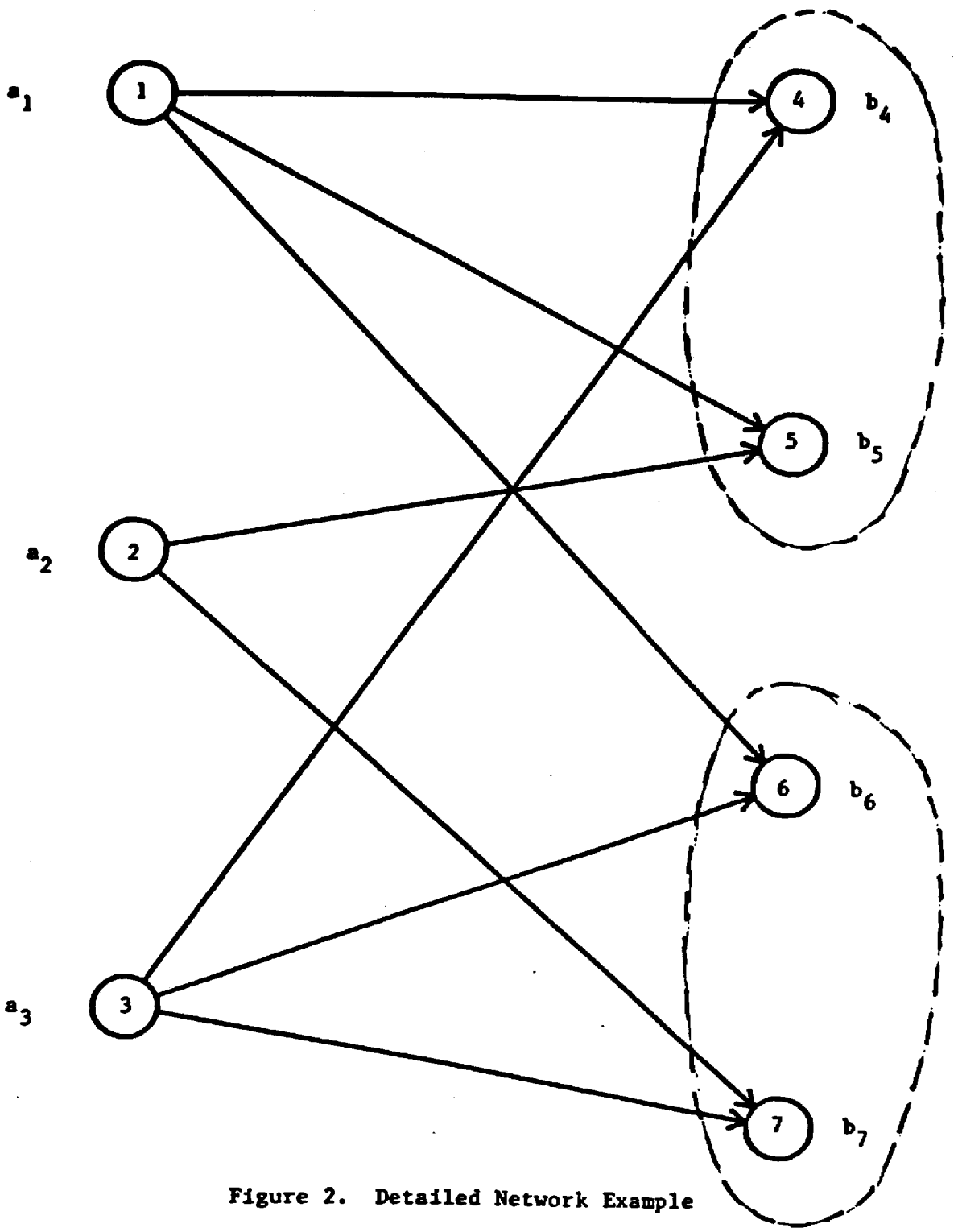


Figure 2. Detailed Network Example

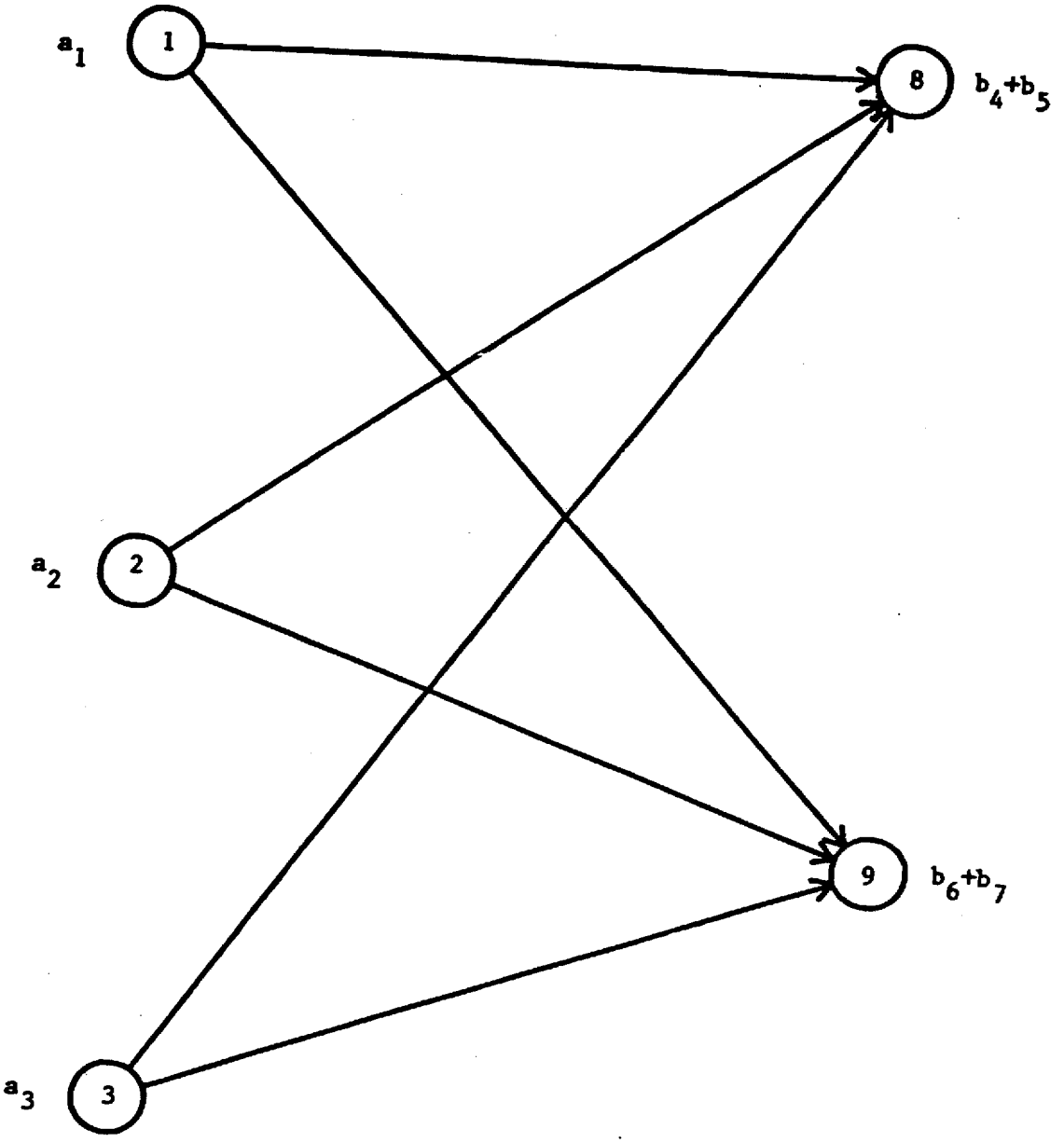


Figure 3. Aggregate Network Example

aggregation. Zipkin also discusses various methods for deriving upper and lower bounds to the solution of the detailed problem at each iteration of the procedure.

Taylor[4] quantified the basic ideas in Zipkin[5] and Geoffrion[2] regarding the choice of components to be aggregated. He establishes a 'closeness' measure between constraints of an linear program. This measure reduces the relation between pairs of constraints to a number between zero and one. It is used to decide which constraints are to be aggregated into clusters, so as to maximize the information available at the aggregate problem level for a given aggregate problem size.

The aggregate problem set up provides a solution which must be disaggregated to provide a solution to the detailed problem. The two basic approaches are as follows

3.1.1 Fixed weight disaggregation

Fixed weight disaggregation essentially multiplies an aggregate solution by a fraction to yield the flows on the detailed arcs. In the case of the complete transportation problem (one with all arcs present between the two node sets) this method can be shown to yield a feasible solution at all times if the multipliers are in proportion to the supply on the incident node as a fraction of the total aggregate node supply.

3.1.2 Optimal disaggregation

The aggregate flows are used to setup independent network flow problems for each cluster. The aggregate flows into each clustered node supply flows to the detailed arcs in the original

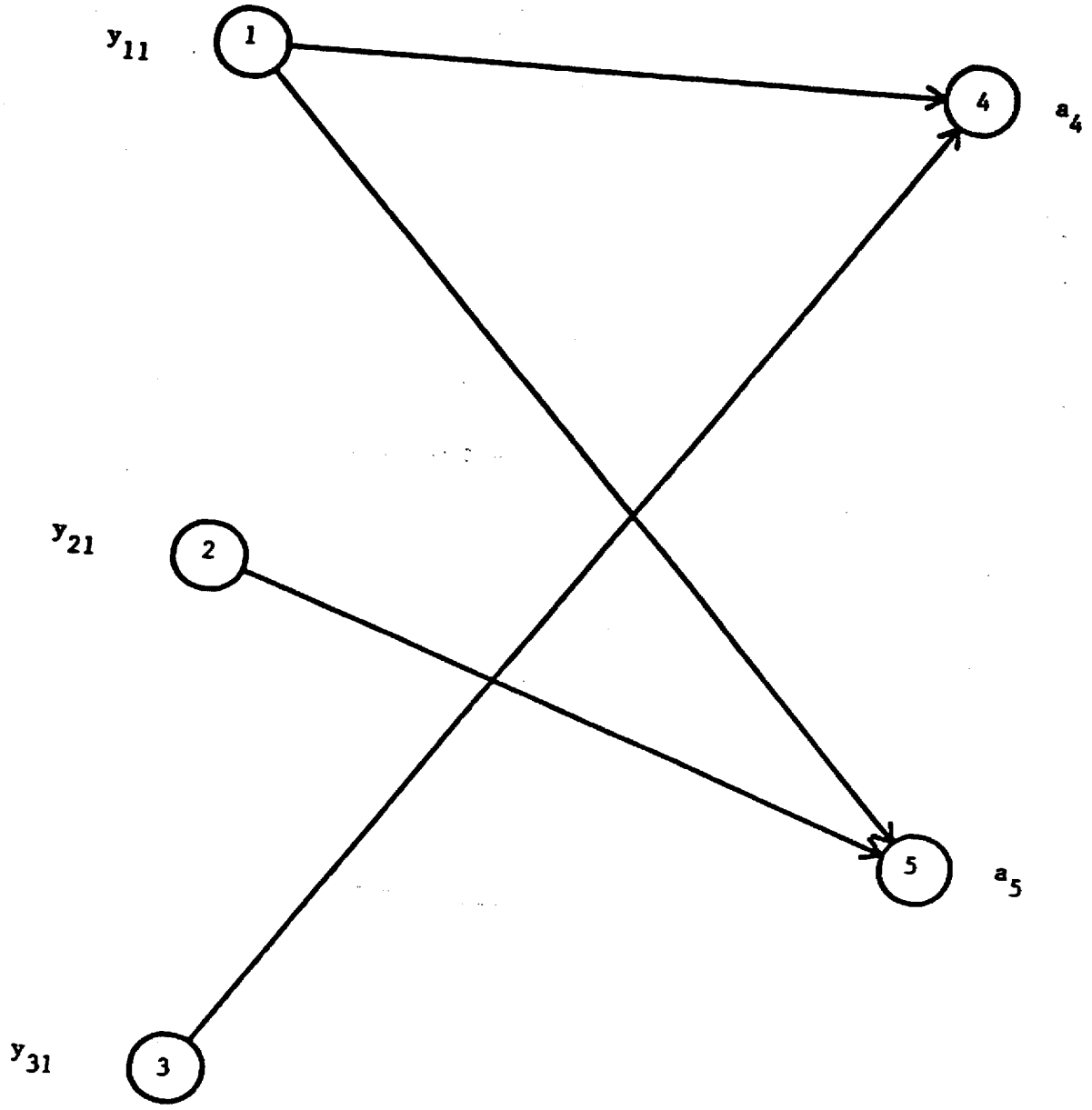


Figure 3(a). Cluster 1 Disaggregation

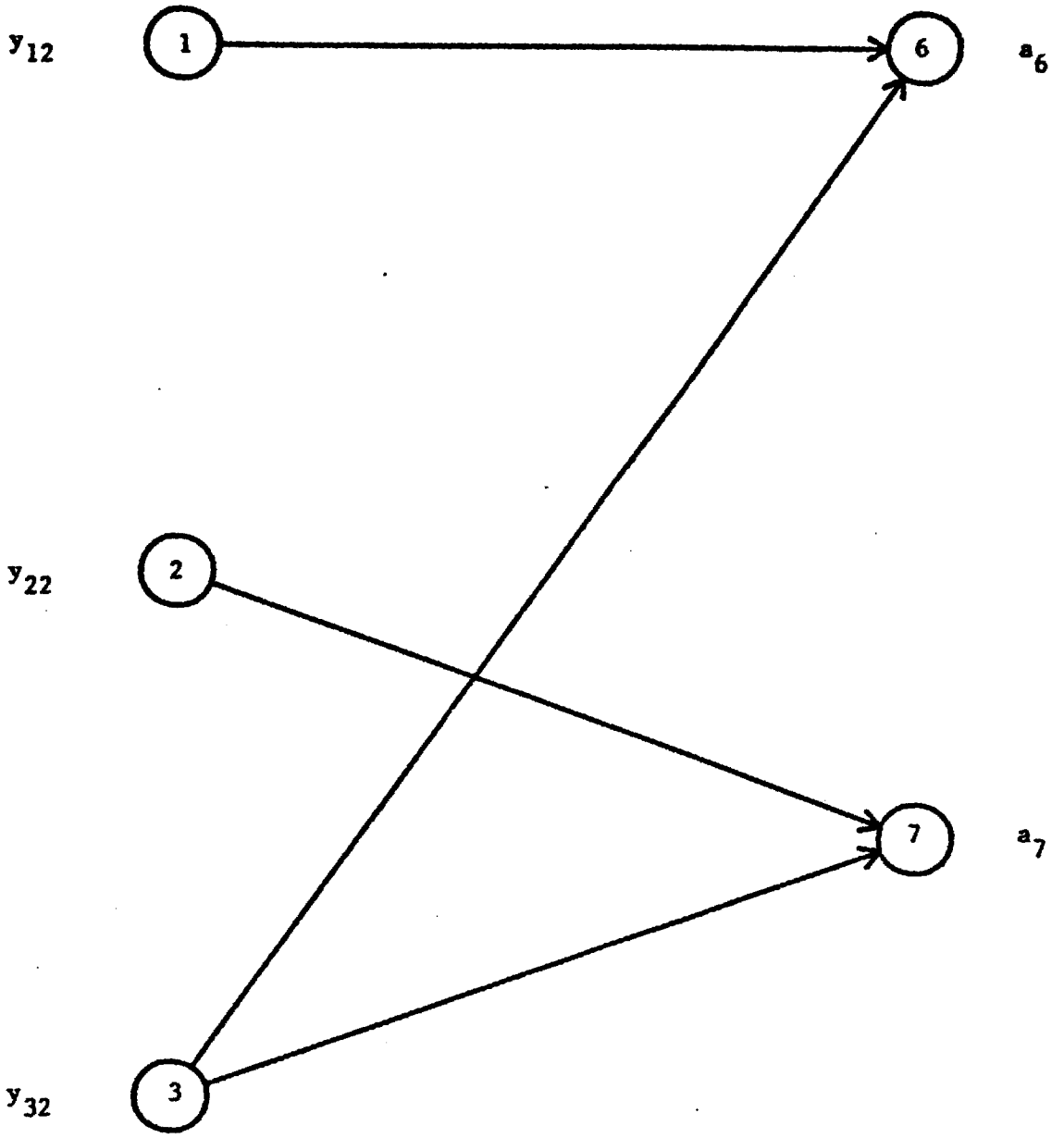


Figure 3(b). Cluster 2 Disaggregation

problem as shown in Figure 2. The effect of the aggregation and optimal disaggregation is to separate the assignment of flows into the detailed arcs in each aggregate node. Arcs in Figure 3(a) are the detailed arcs between the source nodes and the nodes in cluster 1 while arcs in Figure 3(b) are the detailed arcs between source nodes and the nodes in cluster 2.

3.2 Sparsity of the MRMATE structure

When aggregation is used as a solution procedure for the sparse transportation problem, the disaggregated solution cannot be gauranteed to provide a feasible solution to the original problem. A complete transportation problem (with all the arcs present between source and sink nodes) can be assured to be feasible as long as the sum of supplies equal the sum of the demands. This is not, however, true in the general sparse case.

The MRMATE problem with cargo types and different channel types, (i.e. bulk, oversized and outsized categories) is a sparse transportation problem. Assigning outsized cargo to an oversized or bulk channel is an infeasible solution to MRMATE.

The example in Figure 4 shows a sparse transportation problem. A large cost (big-M) is placed on non-existent arcs. Application of Fixed-weight disaggregation in this case is not practical since it ignores the arc costs in disaggregating. It would always try to send flow on the artificial arcs (the ones with cost of big-M). Optimal disaggregation attempts to identify a feasible flow, if it exists. In this case, optimal disaggregation may not identify a feasible solution since the subproblems set up may be infeasible. The infeasibility in this

case results from the fact that the aggregate problem does not have sufficient information about sparsity of the original problem.

Various iterative approaches can be envisaged to handle sparsity.

3.2.1 Adjusting Costs

An approach is to set costs on non-existent arcs. If the costs are too small, the transportation algorithm will not understand that this arc does not really exist. On the other hand, very large costs would tend to inflate the pro-rated cost on the aggregate arc, and may still generate an infeasible solution. An example problem illustrating this case is presented in Fig 4. A large cost (big-M) is placed on the non-exixtent arcs 1-5 and 2-4. The aggregate problem with pro-rated costs is in Figure 5. However, the disaggregation problem may be infeasible. A solution to the aggregate problem which sends a flow of 0 units on arc 1-7 and 15 units on arc 2-7 would set up an infeasible disaggregation problem as in Figure 6.

An algorithm might proceed as follows. Set costs and solve the aggregate problem. If any of the infeasible arcs have flow in the optimal solution, modify the cost just enough to cause a pivot to occur. The principle issue is how to provide for the reduction of arc costs when the arc flow goes to zero.

3.2.2 Adjusting Capacities

Capacities may be established for arcs in the aggregated

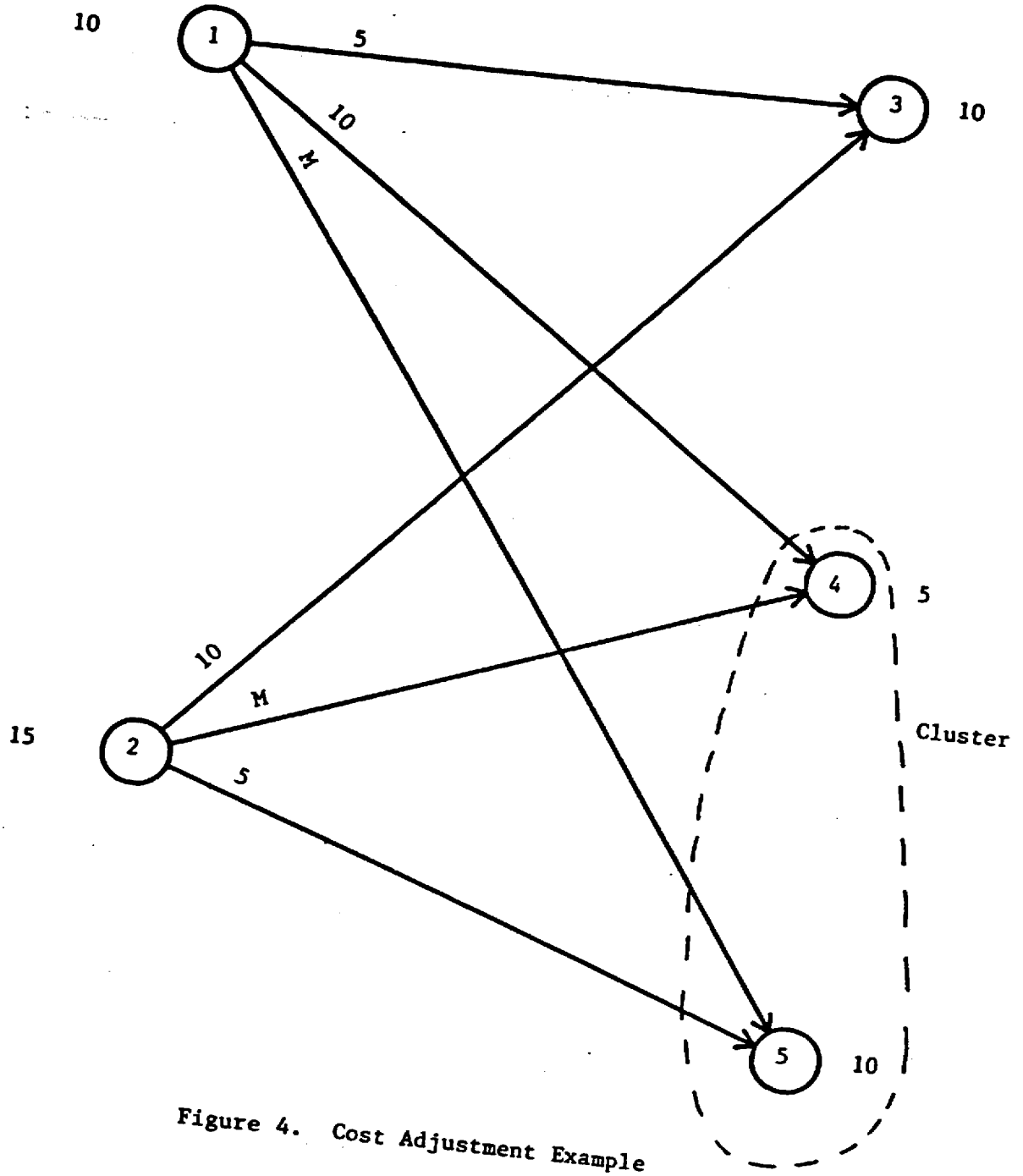


Figure 4. Cost Adjustment Example

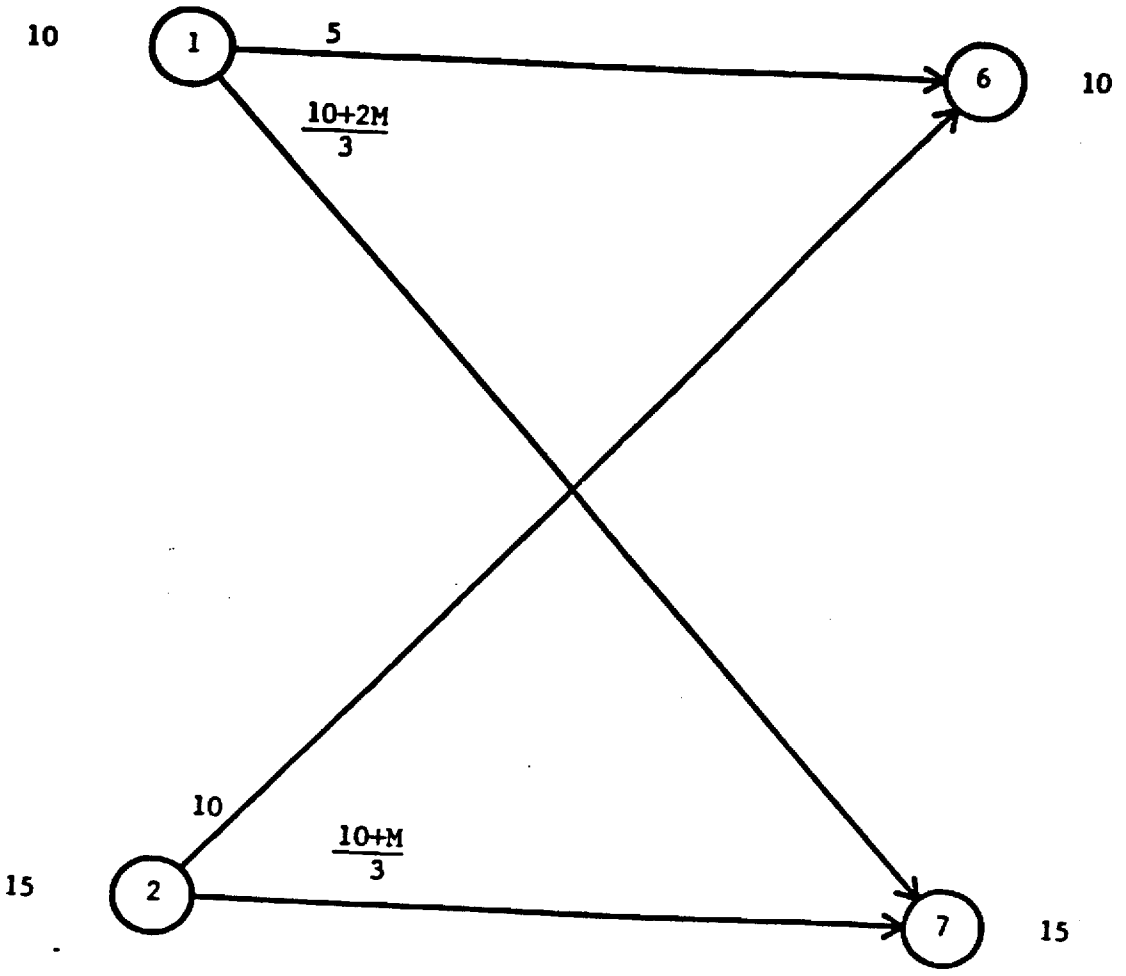


Figure 5. Aggregate Problem with Big-M Costs

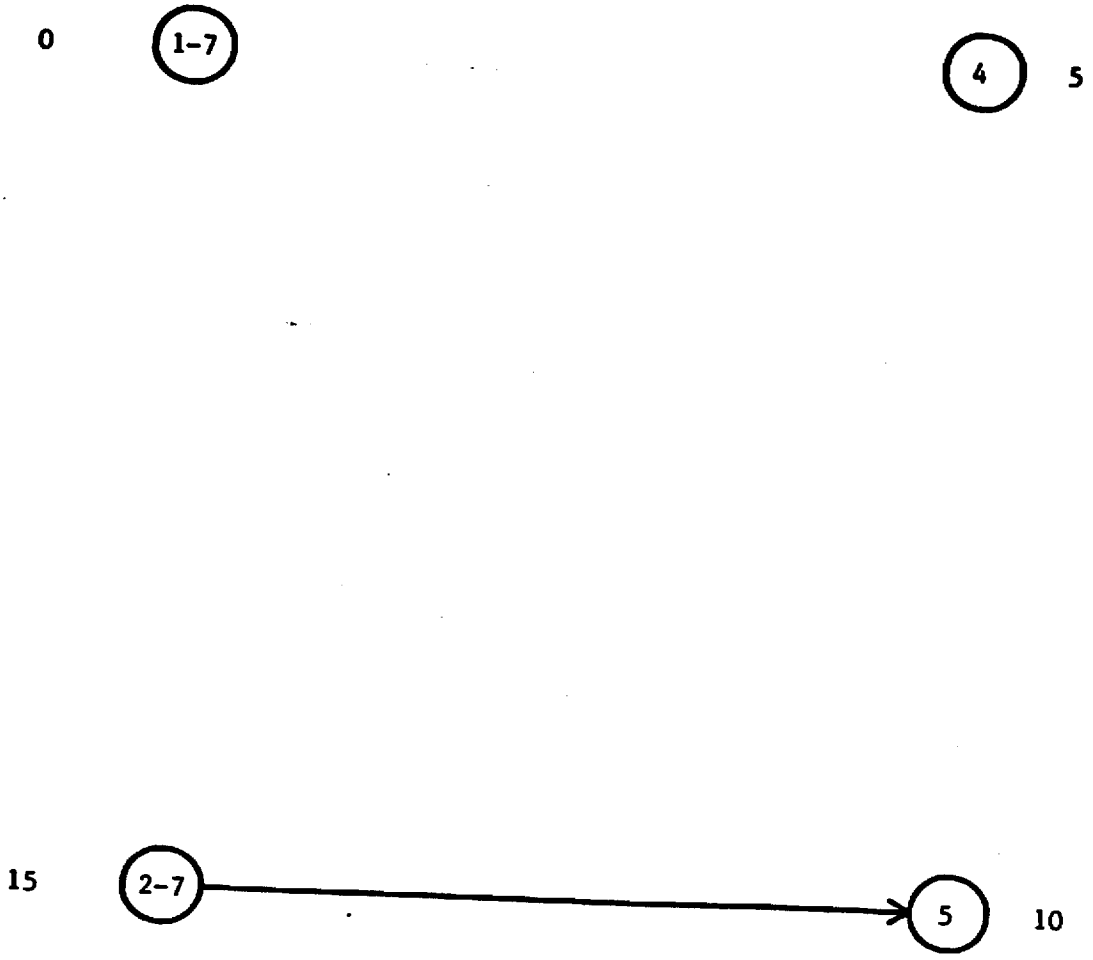


Figure 6. Disaggregation of Cluster Node 7

problem. This problem then becomes a capacitated transportation problem. The capacities may be iteratively modified so as to move towards feasibility of the subproblems and, hence, a feasible solution to MRMATE. A simple approach sets aggregate arc capacity based on the nodes in the cluster to which the source node has arcs. This approach is shown in Figures 7(a) and 7(b). This method can be shown by the example in Fig 8 to still generate infeasible solutions. The aggregate network with arc capacities (as in Figure 7(b)) corresponding to the network in Figure 8 is shown in Figure 9. When two different aggregate arcs have capacity derived from the same node, the aggregate problem loses information regarding which nodes are providing the aggregate capacity. In the example problem in Fig 8, the aggregate arcs from node 1 and node 3 have lost the information that they share their aggregate capacity through sink node 1. A possible solution to the aggregate problem is to send flows of 5 units on arc 1-8 and 15 units on arc 3-8. The disaggregation problem set up in Figure 10 is infeasible due to sparsity of the detailed problem. These arcs cannot all have flow at their upper bounds in the aggregate problem.

3.2.3 Algorithmic Requirement

One of the main requirements is a cohesive format through which all such classes of iterative procedures could be examined. A desirable algorithm to solve the re-formulated problem consisting of the aggregate and disaggregate problems would maintain the network structure of the problems at every iteration while using some mixture of the solutions generated to move

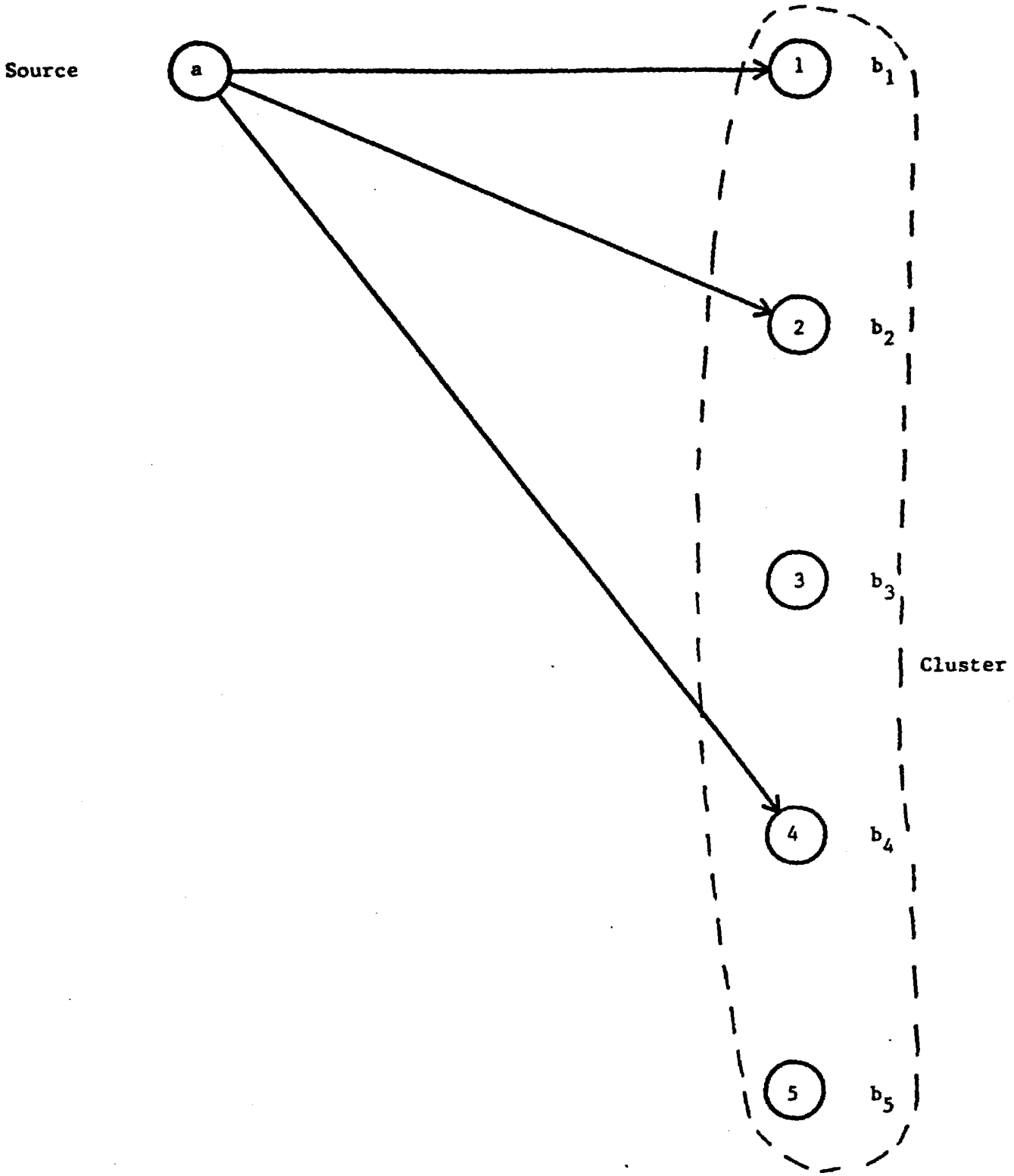


Figure 7(a). Detailed Arcs in a Sparse Network

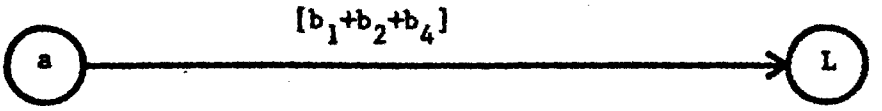


Figure 7(b). Aggregate Arc Capacity

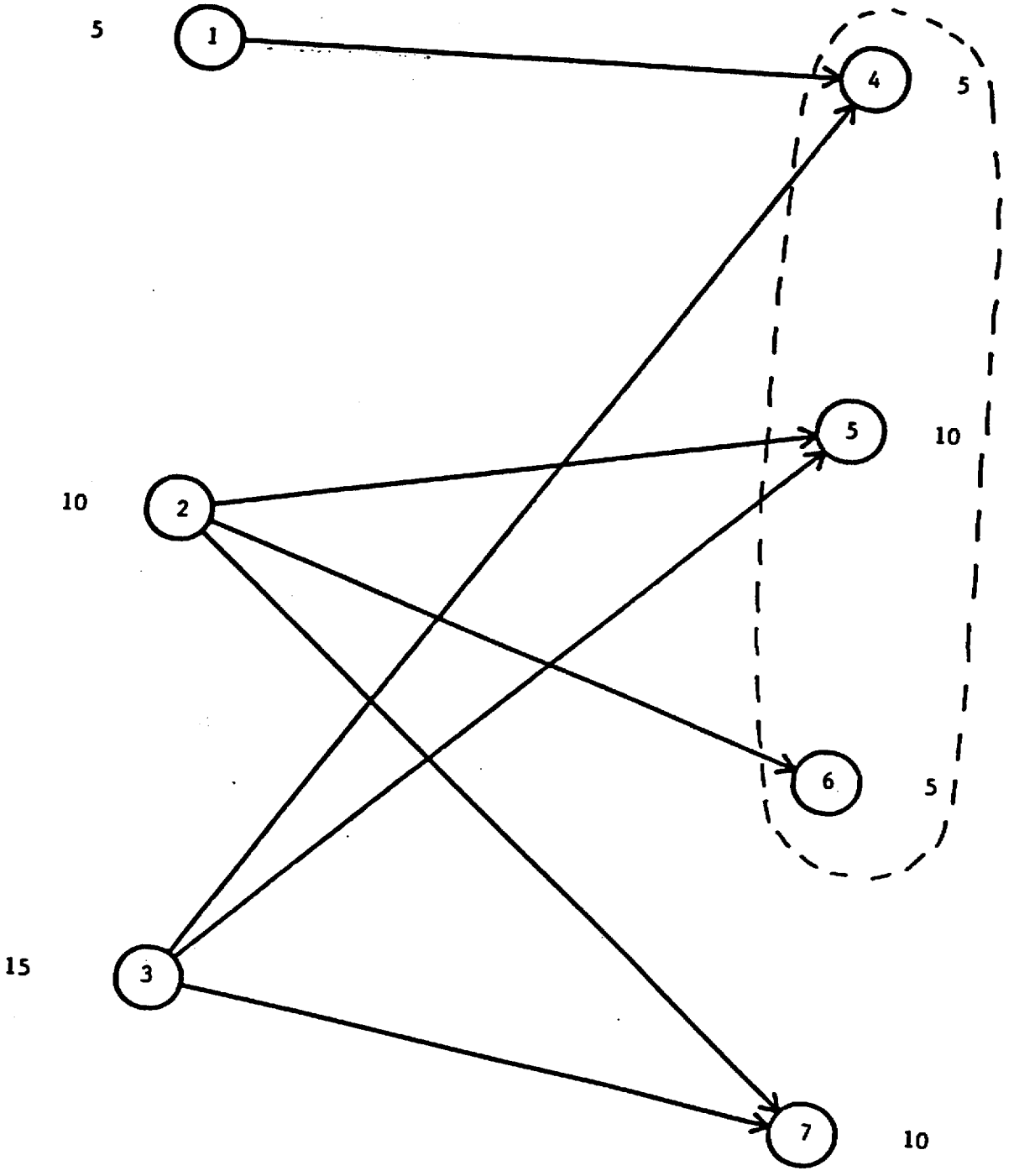


Figure 8. Detailed Network

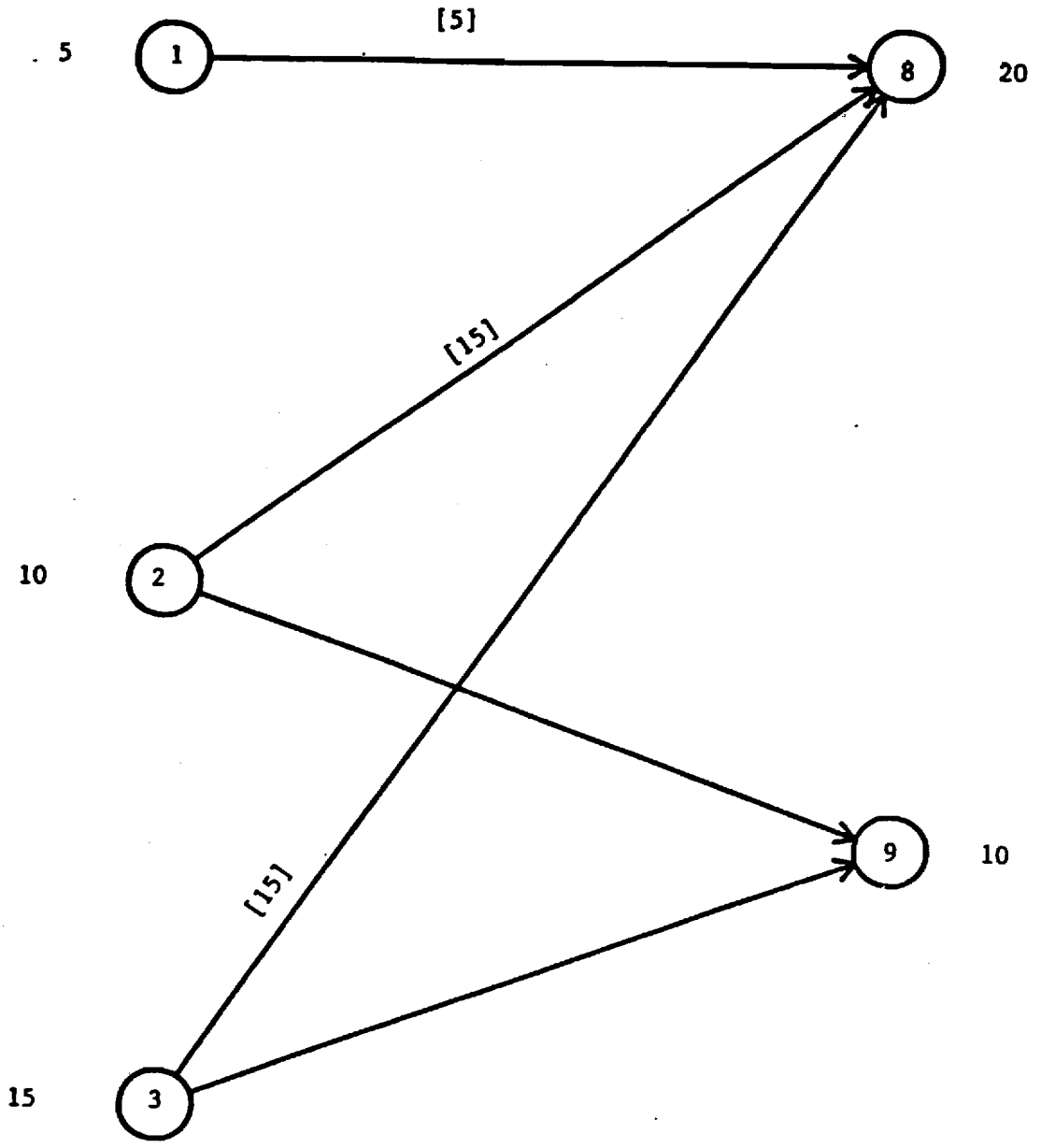


Figure 9. Aggregate Network with Arc Capacities

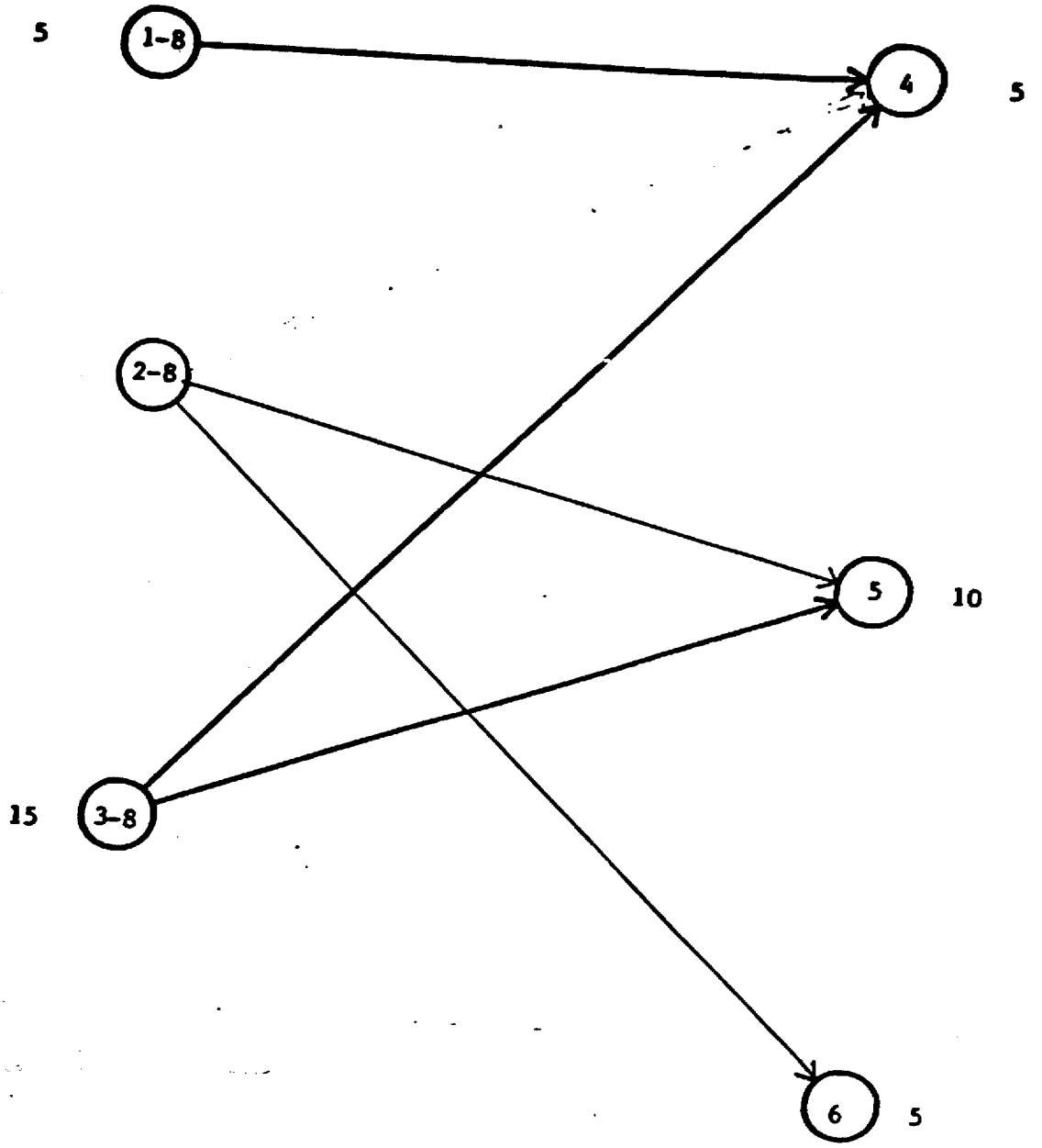


Figure 10. Disaggregation of Cluster Node 8

towards a global optimal solution. A resource allocation procedure in which the aggregate problem sets up resources for the disaggregate problems while the disaggregate problem solutions determine the prices on the aggregate flows would constitute such a desirable procedure.

4.0 TYPES OF MRMATE AGGREGATION

This section discusses different parameters which could be aggregated in MRMATE. Aggregation parameters include aggregation over time, by mode (air/sea), by channels within a time period, and by MRs in a geographical region. An example MRMATE problem is discussed and the effect of different parameter choices on the problem formulation is illustrated.

Consider the MRMATE example in Figure 11. The planning horizon is two periods long. There are four MRs and four channels. The source nodes are the four MRs with the demands equal to the force quantity. The sink nodes are the time expanded channels, hence there are 4 x 2 = 6 sink nodes. The channel capability is constant over the planning horizon, so the sink node supplies are determined. MR 1 has to be available on day 1 while MR 2 is available at the channel only on day 2, this information implies that there are no arcs from MR 1 to channels in time period 2 and from MR 2 to channels in time period 1. Also the cargo categories of the MRs permit MR 1 to be shipped on channel 1, MRs 2 and 3 on channels 1 and 2, and MR 4 on any of the channels. The assignments of cargo types to feasible channels at available time periods determines the arcs in the MRMATE network in Figure 11. Also, channels 1 and 3 are given to be air channels, channels 3 and 4 as sea channels. Since the capability of air channels is a weight constraint, that of sea channels is a volume constraint while the MR quantities are expressed in weight units, there is a multiplier associated with each arc to convert the flows to the proper units. The multipliers on arcs to air channels is 1 because the supply and

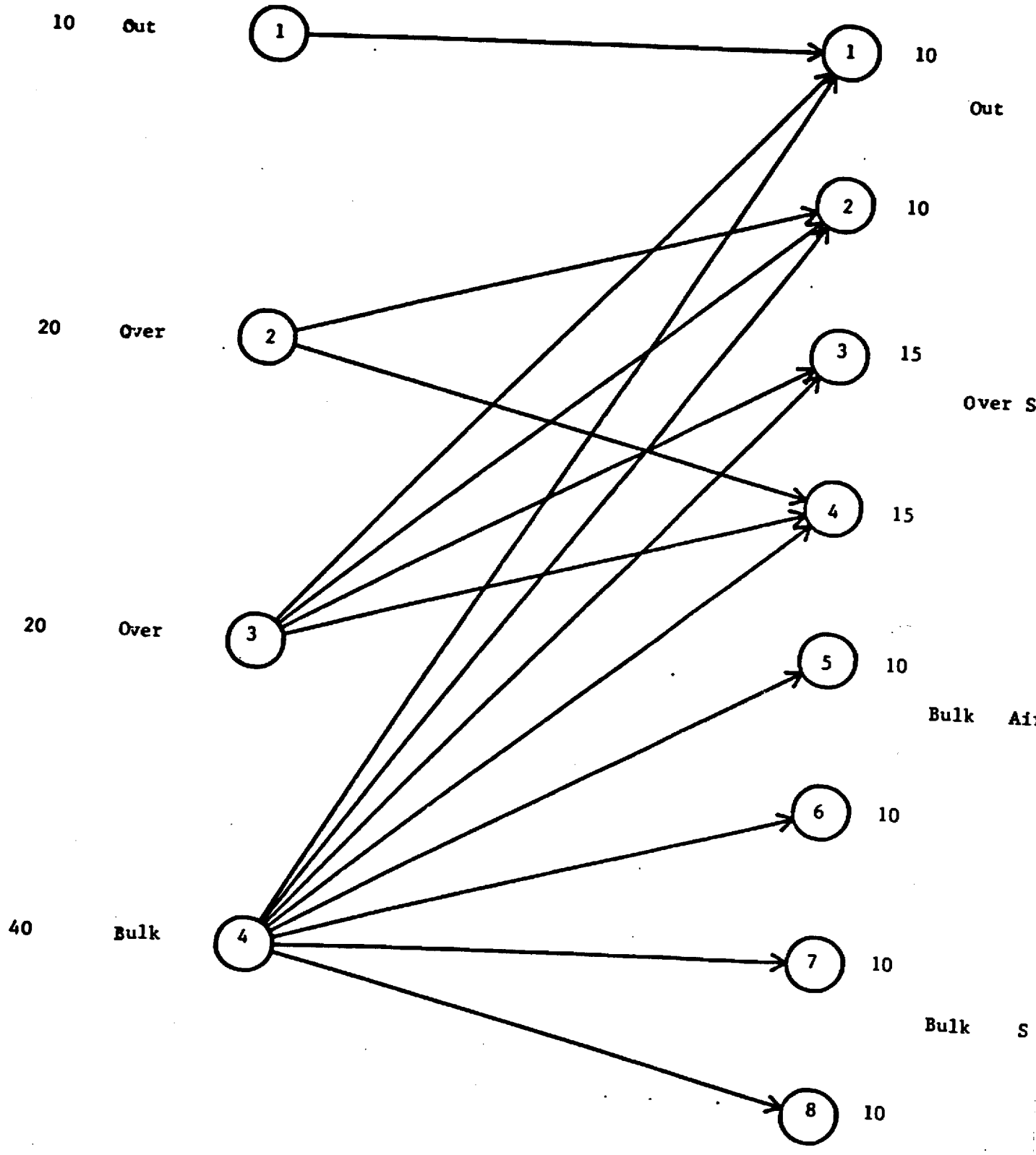


Figure 11. MRMATE Example

demands are in the same units. The multipliers for the four MRs are 1,2,3,2 respectively.

4.1 Aggregation over time

In MRMATE, planning periods and time windows are used to setup channel capabilities over time, and to construct costs of assigning a MR to a channel at a certain time t. Aggregation over time involves creating an aggregate problem by combining information over time. The aggregate problem would have a network structure with the supplies, MRs, connected by arcs to feasible channels based on cargo type. Channel capability is T * per-unit-time channel capability for each channel. At the aggregate level, time information is absent and the problem becomes one of assigning the MRs to feasible channels. Thus the aggregate problem has

Number of sources = Number of MRs

Number of sinks = Number of channels with non-zero flow as generated by LIFTCAP

Max number of arcs = M * A * I * J

where I = number of POEs and J = number of PODs.

The disaggregation problems schedule the MRs, allocated to a channel, across time. The data used to set up these problems are time window information of each MR (providing arc costs) allocation of MRs to the channels at the aggregate level (providing supply information and allocation of channel capability across time). This aggregation procedure can be interpreted as the process of separating the detailed scheduling

information from the assignment decisions of MRs to channels.

There are (I*J) disaggregation problems set up. The aggregate level problem is a generalized network flow problem; the disaggregation problems are pure network flow problems.

It is likely that solving such a series of smaller problems, even though one of them is still a generalized network flow model, will yield smaller overall MRMATE solution times.

4.1.1 Example Problem

For the example problem in Figure 11, aggregation over time yields the aggregate problem as in Figure 12. Arcs are introduced between a MR and an aggregate node if there is atleast one detailed arc between that MR and any one of the detailed nodes in the cluster of nodes forming the aggregate. Since MR 1 can be allocated only to channel 1, there are no arcs between MR 1 and any of the other aggregate nodes. The aggregate node capabilities are the sum of the detailed node capabilities as indicated in Figure 12. Since the multiplier associated with all nodes in a cluster are the same, the aggregate arc has the same multiplier as the detailed arcs.

An aggregate problem solution is used to setup the disaggregate problems as in Figures 13(a)-(d) by multiplying the flows by the appropriate multiplier. Thus the aggregate flow between MR 2 and channel 2 is multiplied by 2 to setup the demand in the disaggregate problems in the same units as the detailed node capabilities. Thus detailed channels in disaggregation problems 1 and 3 in Figures 13(a) and 13(c) would have weight units while those in disaggregation problems 2 and 4 in Figures

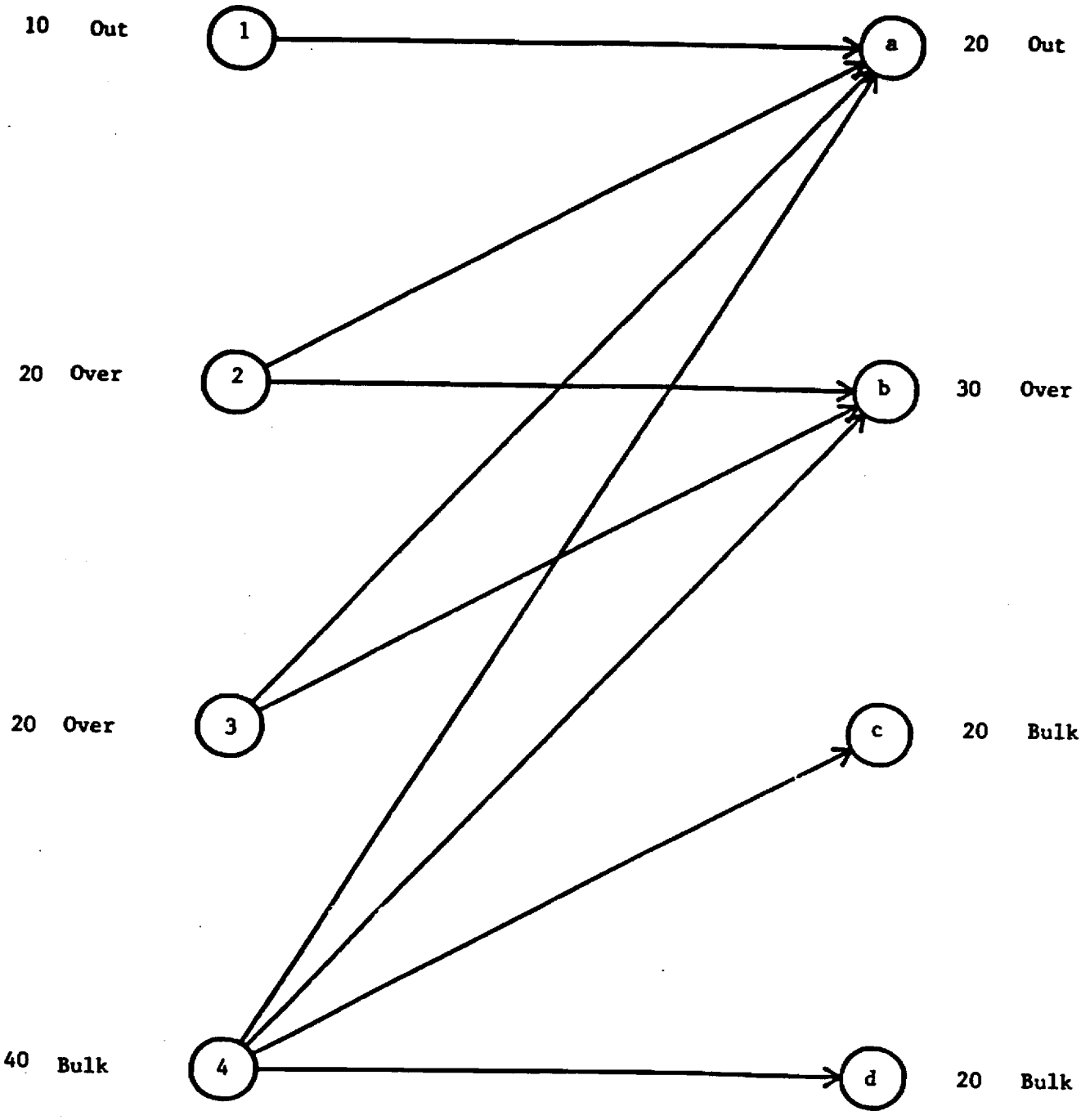


Figure 12. Aggregation over Time

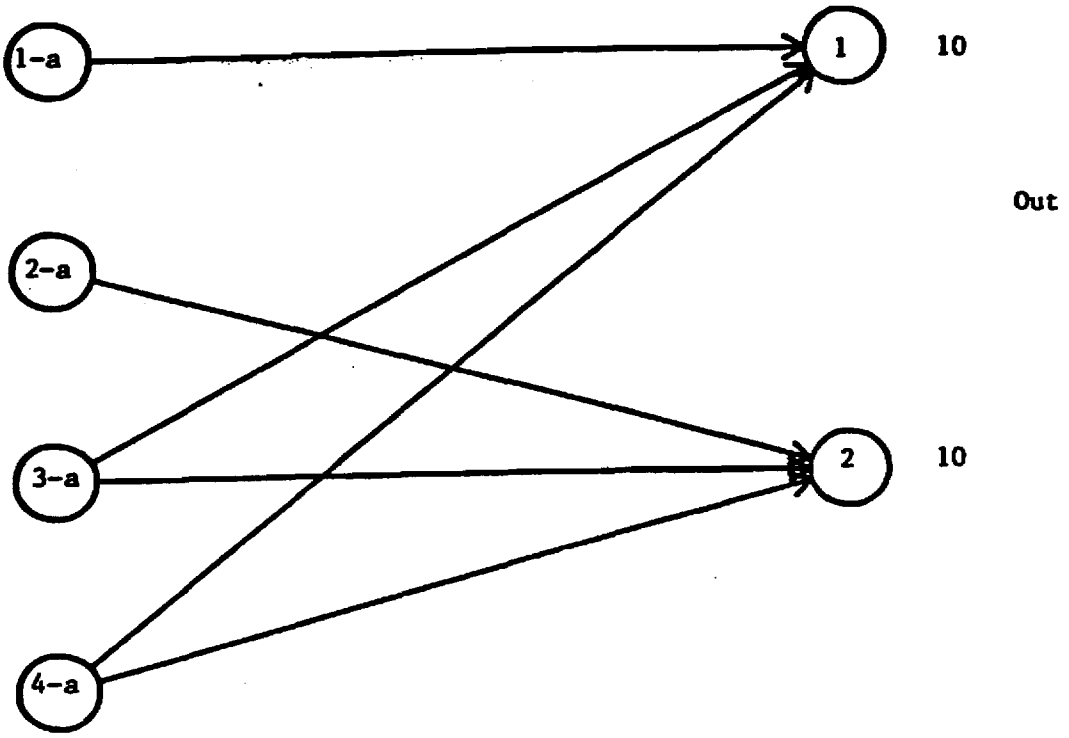


Figure 13(a). Channel 1 Disaggregation

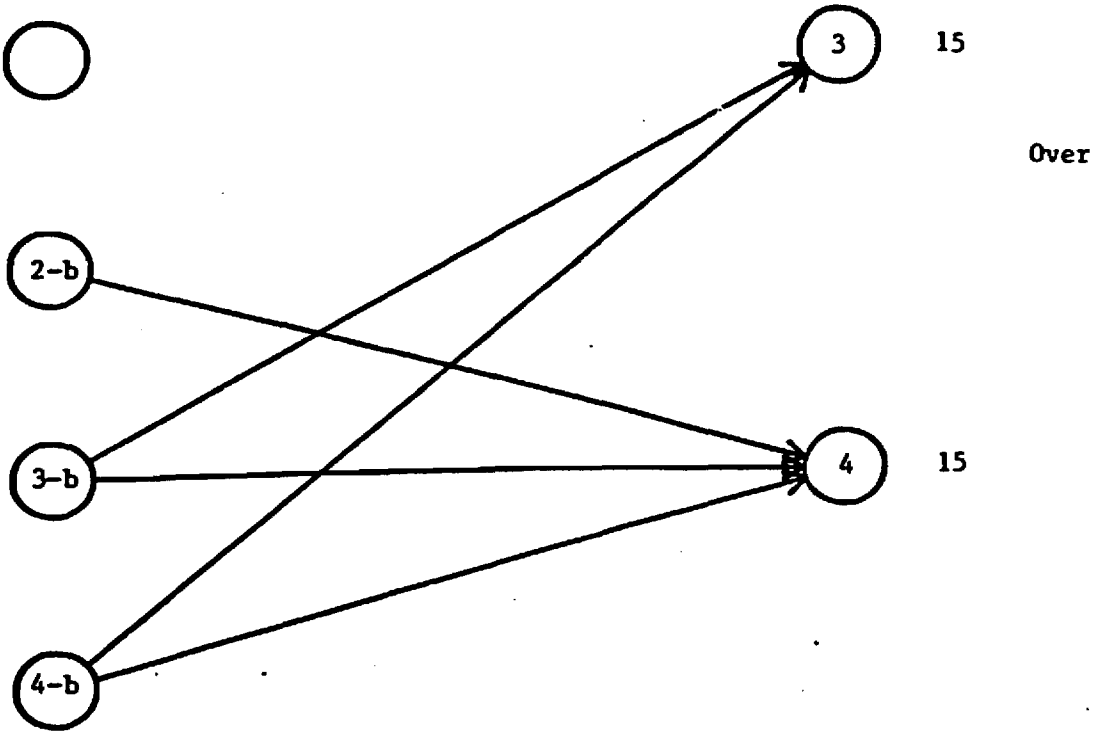


Figure 13(b). Channel 2 Disaggregation

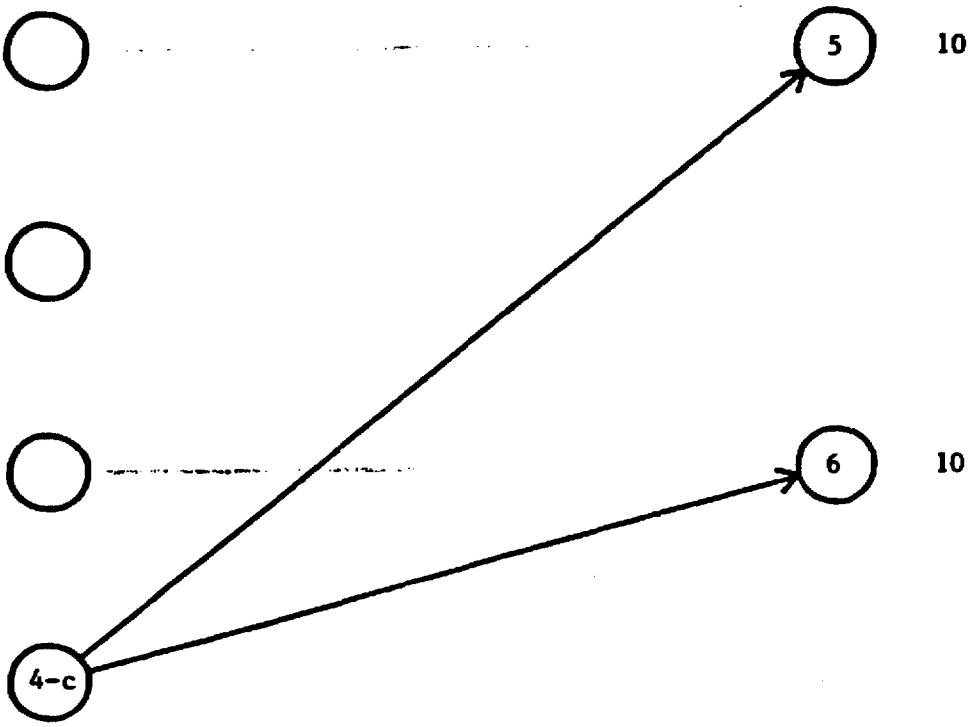


Figure 13(c). Channel 3 Disaggregation

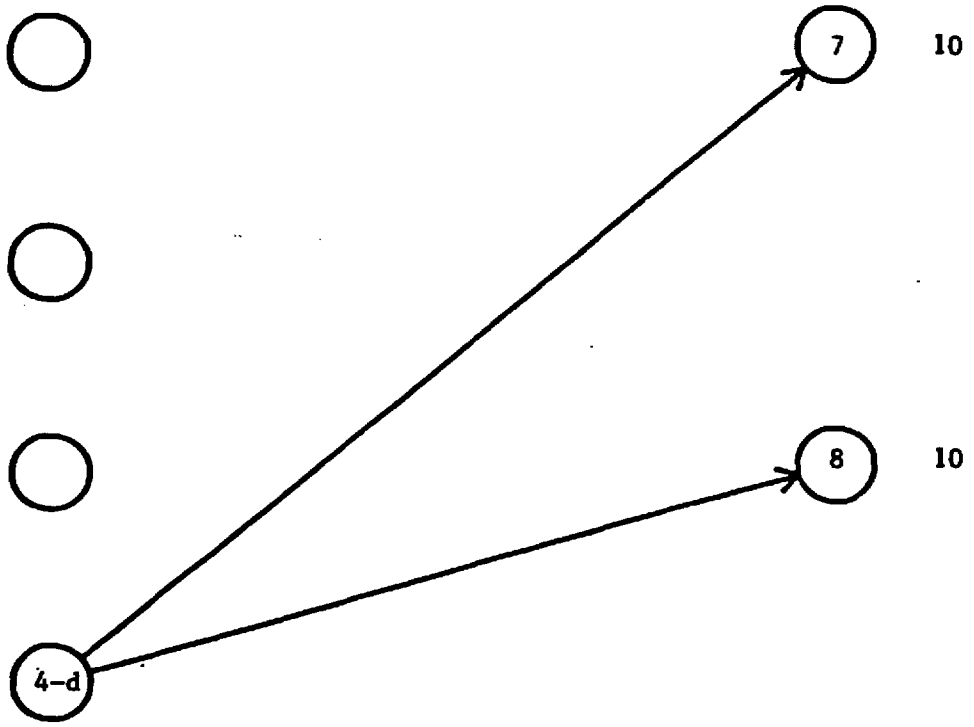


Figure 13(d). Channel 4 Disaggregation

13(b) and 13(d) would be in volume units.

4.2 Aggregation by mode

A second decomposition of MRMATE into a hierarchical structure is provided by aggregating channels which employ a given mode (i.e. air/sea).

The aggregated problem attempts to allocate MRs to each of the two aggregate channels representing air and sea modes. Information on MRs are their supplies. The demand on the two modes is the total capability of all channels across the entire planning period employing the mode. The decision at the aggregate level is the mode split. Information, such as the time windows of MRs or detailed channel capabilities within a mode, is ignored at the aggregate level.

The disaggregation problems make decisions regarding detailed scheduling and allocation within each mode (air/sea). Since the multipliers on arcs for a given mode are the same, the two problems are pure network flow problems. At the disaggregation level, time window information provides costs on the arcs; channel capability and planning period data determine the demand information. Supplies represent the portion of MRs allocated to a mode and are given by the product of aggregate flows and the multiplier for the mode.

This aggregation structure has the added benefit of isolating elements in the problem which affect changes only in a particular mode. Also, since the aggregate problem is a generalized flow problem with only two demand nodes, very fast

procedures could be used to produce a solution. Finally, the disaggregation problems are pure network problems, and this also speeds up the overall solution time.

4.2.1 Example Problem

Aggregation by mode applied to the example in Figure 11 yields the aggregate problem in Figure 14. The aggregate channel capabilities are set up as before. Since all the arcs joining a MR to channels of the same mode have the same multiplier, the aggregate arc multipliers are the same as the multipliers an the detailed arcs. The aggregate problem in Figure 14 is a generalized network flow problem.

Disaggregate problems are set up for each mode as in Figures 15(a) and 15(b). The aggregate flows are multiplied by the appropriate MR multiplier for the sea problem in Figure 15(b), while the air problem uses the aggregate flows in the same units. The disaggregate problems are pure network flow problems.

4.3 Aggregating channels within a time period

Planning periods in MODES refer to periods of time over which channel allocation of assets remain constant. Dividing the planning horizon into T periods has the effect of increasing the number of demand nodes by a factor of T. Also, planning periods infer simultaneous assignment and scheduling decisions in MRMATE. It would be useful to explore aggregation models which separate these two decisions. For example aggregating all channels of same mode in each time period corresponds to making a scheduling decision at an aggregate level and an assignment decision at the

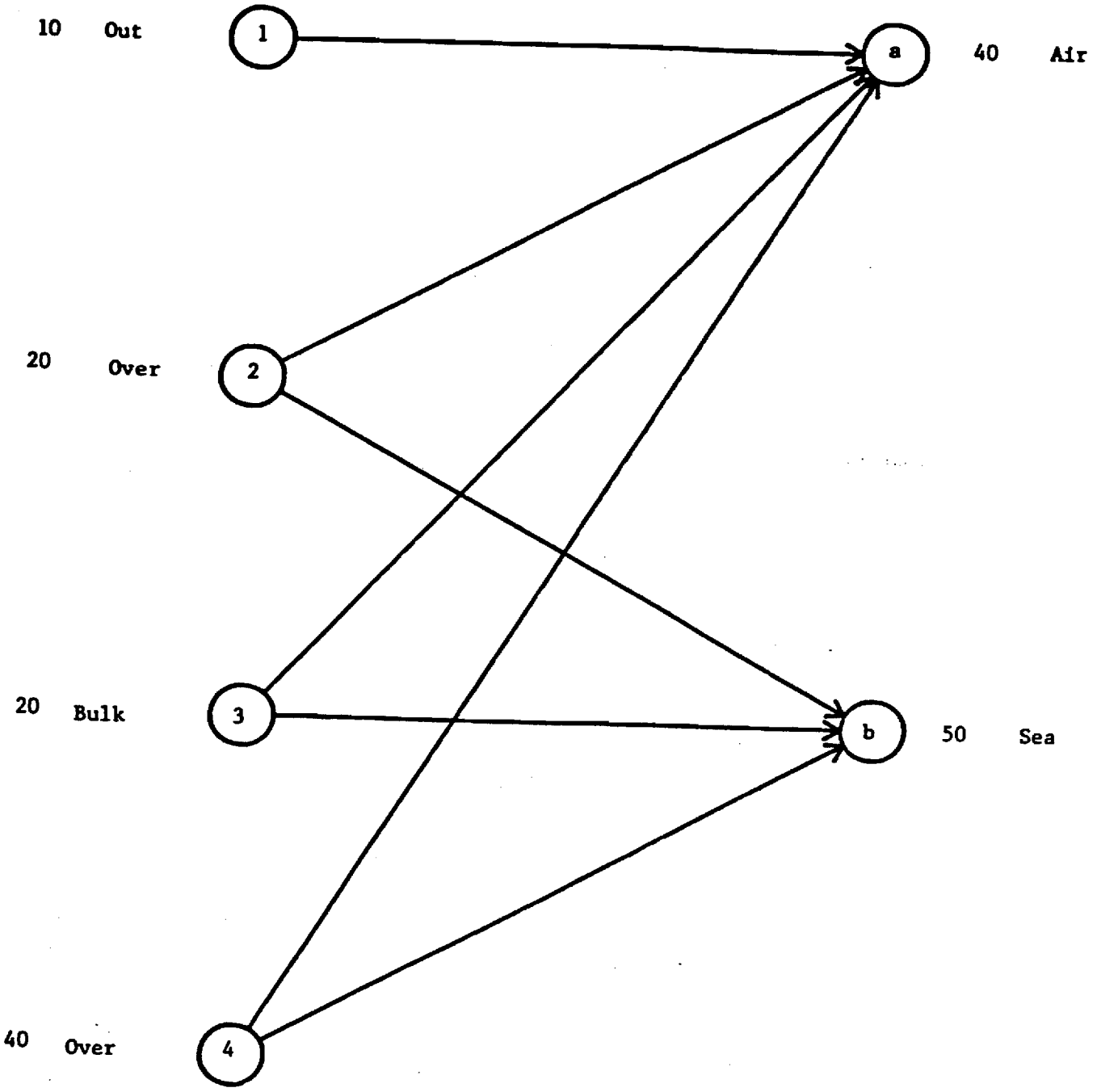


Figure 14. Aggregation by Mode

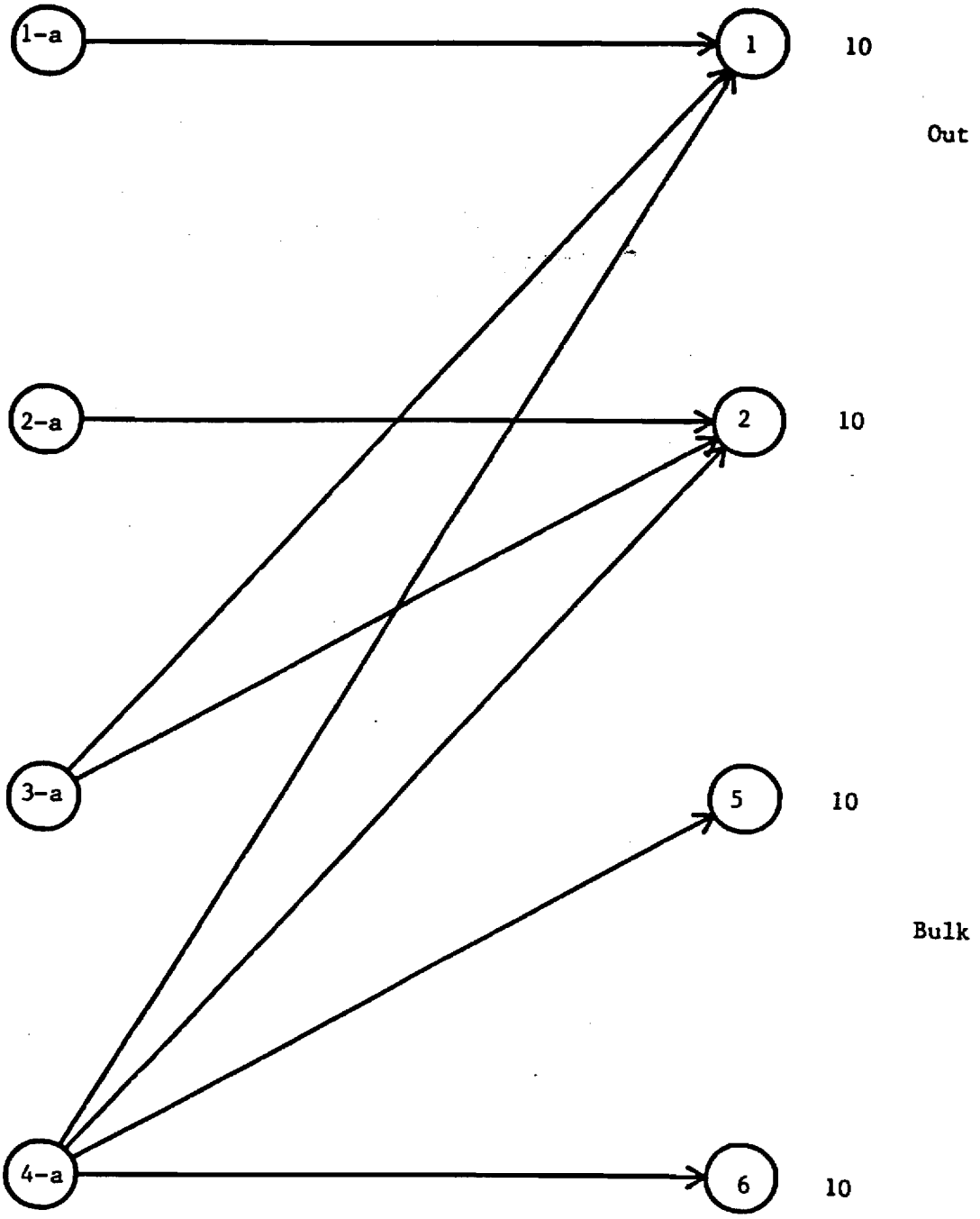


Figure 15(a). Disaggregation Problem for Air Channels

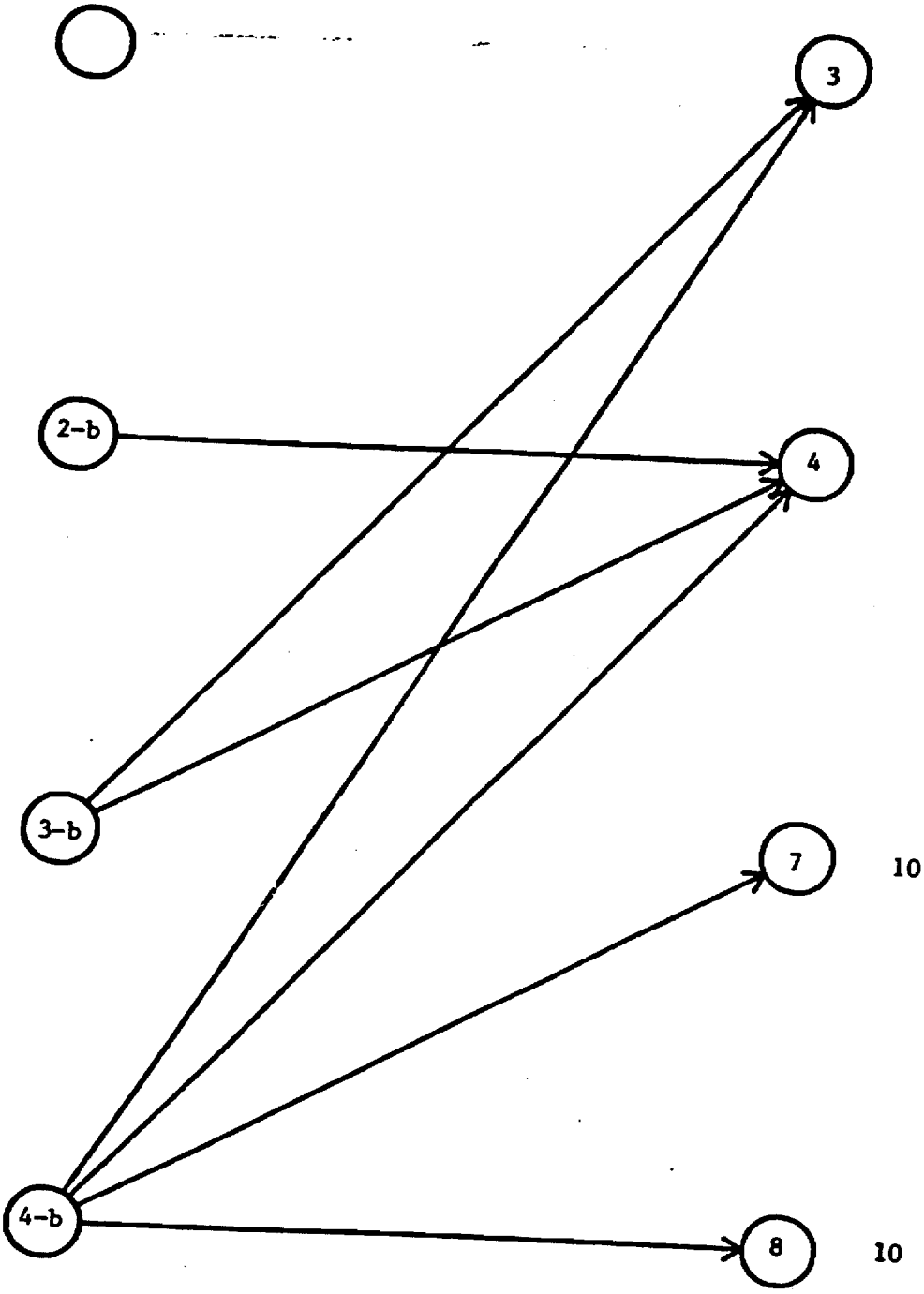


Figure 15(b). Disaggregation Problem for Sea Channels

detailed level. This is the reverse of the decision making roles described in Section 5.1.

The aggregate problem decides which time periods a MR be allocated to. Time window information for each MR is used to setup costs on the aggregate arcs.

At the disaggregate level, 2 * T problems are solved to assign MRs shipped during each period to the available channels in a period. This process could be viewed as a process of deciding on a shipping schedule for the MRs for each day of the planning period.

Again the aggregate problem is a generalized network flow problem while the disaggregation problems are pure flow problems. Also, this formulation enables additional constraints, such as unit integrity, to be applied at the second level, i.e. on a day to day basis.

4.3.1 Example Problem

Aggregating channels of the same mode within time periods in the problem in Figure 11 yields an aggregate problem as in Figure 16. Since the channels that are aggregated are of the same mode type, the aggregate arcs have the same multipliers as the detailed arcs. The aggregate problem is a generalized flow problem.

Given the aggregate problem solution 2 * 2 = 4 disaggregate problems would be set up. The disaggregate problems 1 and 2 in Figures 17(a) and 17(b) would use the aggregate flows in the same units (air multiplier = 1), while in disaggregate problems 3

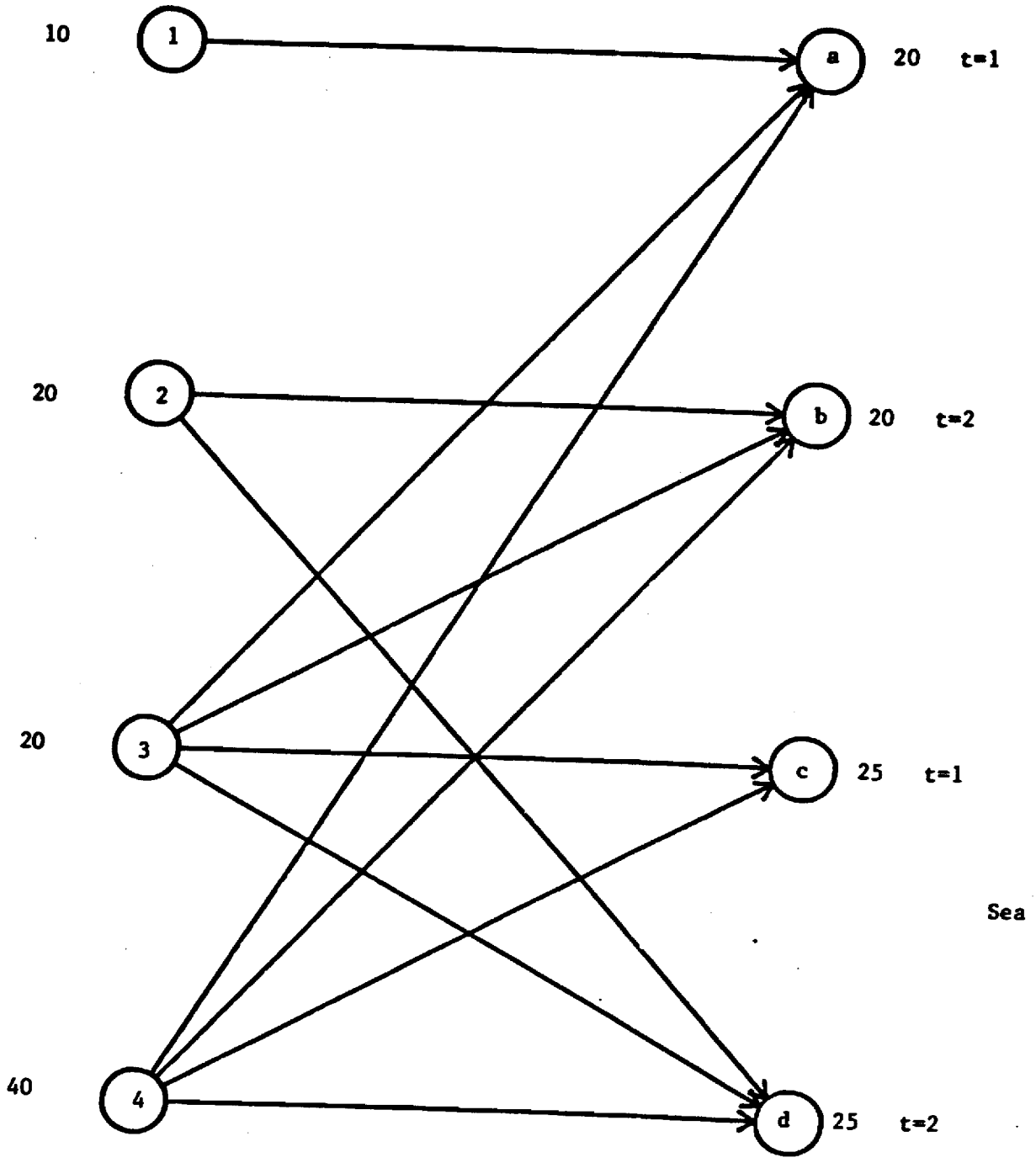


Figure 16. Aggregating Channels by Mode within a Time Period

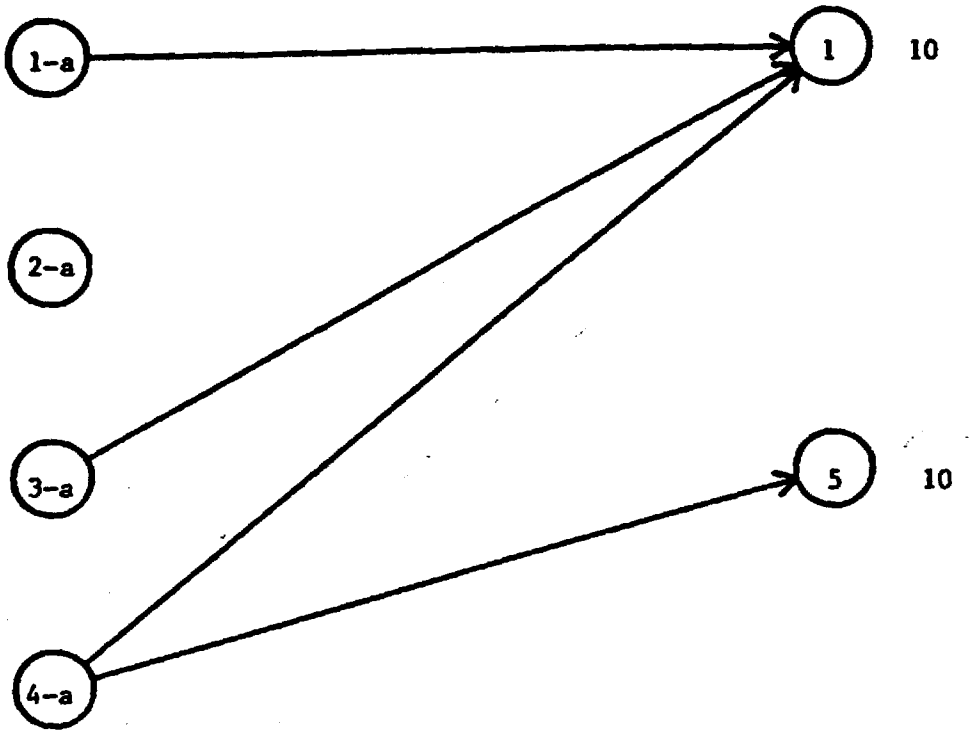


Figure 17(a). Disaggregation Problem for Air at t=1

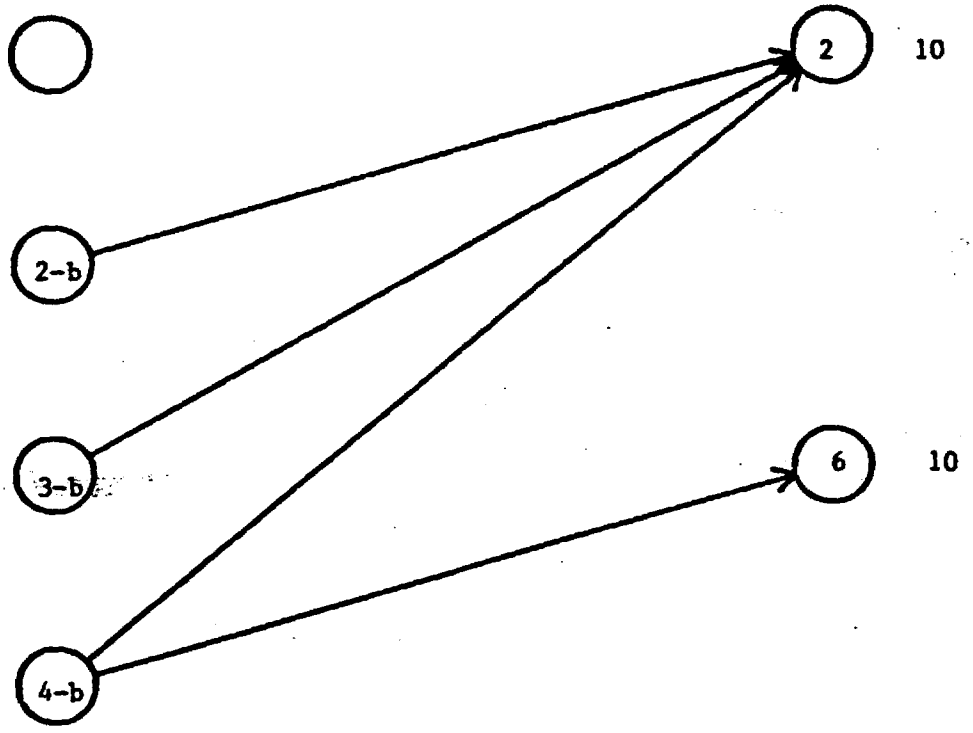


Figure 17(b). Disaggregation Problem for Air at t=2

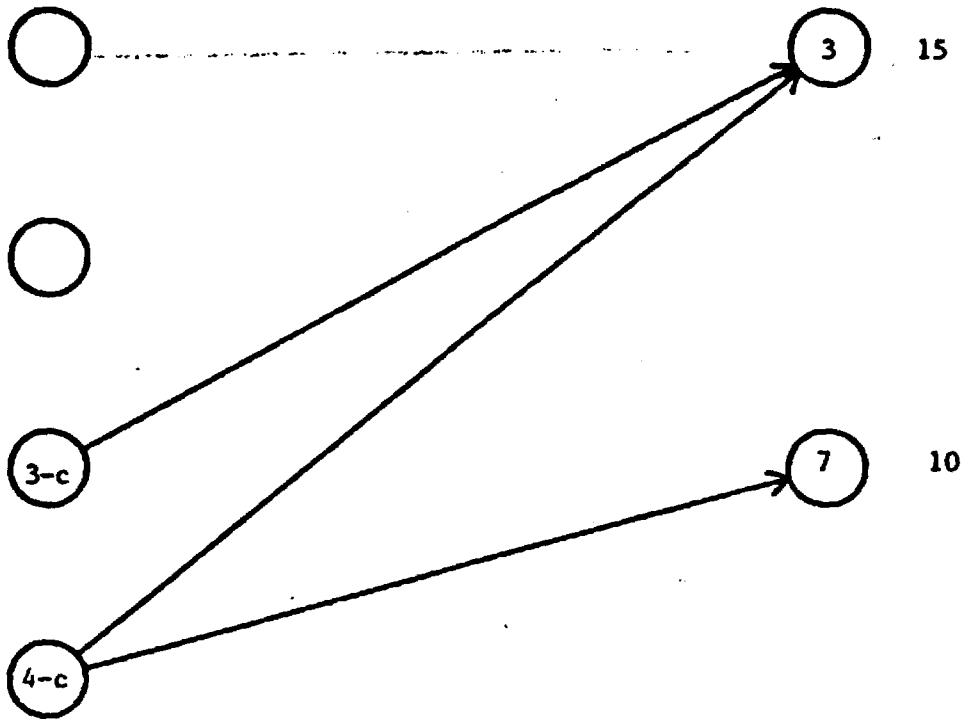


Figure 17(c). Disaggregation Problem for Sea at t=1

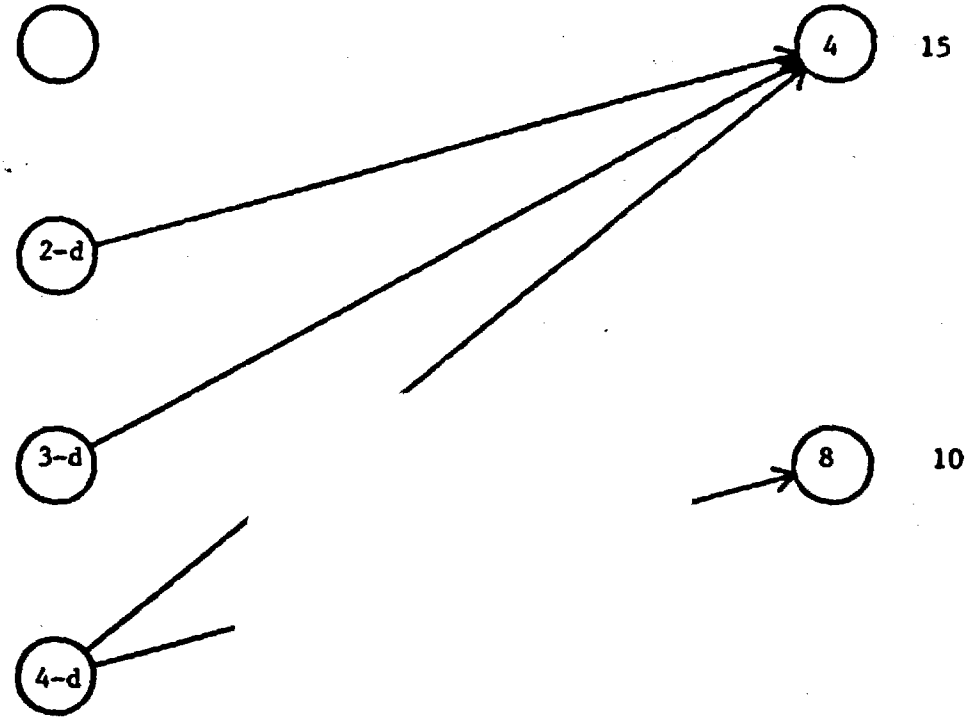


Figure 17(d) or Sea at t=2

and 4 in Figures 17(c) and 17(d) the flows are multiplied by the appropriate MR multiplier.

5.0 CONCLUSIONS

This report discussed logical aggregation parameters in MRMATE. An MRMATE model formulation was proposed which would utilize these parameters to set up a hierarchical decision making model. Such a model would enable separation of disjoint decision making regions, reduce problem sizes, and facilitate human interface.

Solution techniques for such models are being developed which would maintain the model structure at each level, while moving towards the global optimal solution to the overall MRMATE problem.

6.0 BIBLIOGRAPHY

1. Balas, E., 'Solution of Large-Scale Transportation Problems through Aggregation', Operations Research, 13, pg 82-93, '65.

2. Geoffrion, A., 'Priorl Error Bounds for Procurement Commodity Aggregation in Logistics Planning Models', Naval Research Logistics Quarterly, 24, pg 201-212, '77.

3. Lee, S., 'Surrogate Programming Through Aggregation', Ph.D. Dissertation, UCLA '75.

4. Taylor, R.W., 'Aggregate Programming in Large Scale Linear Systems', Ph.D. Dissertation, Georgia Tech., '83.

5. Zipkin, P.H., 'Aggregation in Linear Programming', Ph.D. Dissertation, Yale University, '77.

Report for:
Joint Deployment Agency
MacDill Air Force Base, FL 33608

MODIFICATIONS TO SEALIFT CAPABILITY ESTIMATION IN MRMATE

John J. Jarvis
Kevin L. McCroan
William G. Nulty
H. Donald Ratliff
Michael A. Trick

PDRC 86-06

Report by:
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

This work is supported by the Office of Naval Research under Contract No. N0014-85-C-0797 and N00014-83-K-0147. Reproduction in whole or part is permitted for any purpose of the U.S. Government

TABLE OF CONTENTS

1.0 INTRODUCTION

The System for Closure Optimization Planning and Evaluation (SCOPE), developed by Georgia Tech's Production and Distribution Research Center, was designed to solve deployment problems having many movement requirements and many assets with reasonable accuracy. Complete accuracy was deemed unattainable because of constraints on a) time to develop the model, b) time to solve the model, and c) existing technology to design and implement such a model.

A simplifying assumption was made which satisfied those constraints. This assumption, that assets could be considered to maintain constant and continuous flows across channels, allowed lift capability of both air and sea channels to be described by a constant flow rate (e.g., stons per day) during any planning increment. Good accuracy could be achieved for most problems likely to occur, with greater accuracy achieved as problem size increased. The ability to achieve very accurate solutions to small problems, such as those that could be reasonably solved by hand or by inspection, was considered of lower importance.

Analysis of the model indicates that accuracy is a function of numbers of assets, transit times, cycle times, load (and unload) times, and effective planning horizons. For large problems, where the number of assets is large and/or the quantity of movement requirements results in a large effective planning horizon, the inaccuracies are small. For smaller problems, inaccuracies for air assets are still generally small, since air channel data is of a smaller order of magnitude than the accuracy

desired (hours vs. days). For sea channels, however, the resulting errors may be unacceptable, since channel data and model answers are of the same order of magnitude.

Georgia Tech has undertaken a three-phase plan of study to modify SCOPE so that more accurate closure estimates for sea channels may be provided. These phases of study are briefly described in section 1.2, and the first phase is fully described later in this report.

1.1 Continuous Flow of Assets

Since it may not be clear what is meant by continuous flow of assets, a brief explanation is needed. When assets continually cycle between the same POE/POD pair, the capability of that channel (POE/POD/asset-type combination) may be loosely described by a rate of flow (e.g., stons per day). This rate is given by

$$R = C / T$$

where

R = Channel capability (e.g., mtons per day)

C = Total capacity of all assets applied to the channel (e.g., mtons)

T = Cycle time (e.g., days).

Cycle time represents the time between consecutive visits of a particular asset to a POD, i.e., the time from POD to POE to POD.

While a specification of a rate of flow for a particular asset during some small interval of time (that is, small relative to cycle time) may obviously be inappropriate, since we may be interested only in the number of deliveries actually completed

during that interval, it should also be obvious that the longer the time interval considered (or equivalently, the shorter the cycle time), the more accurate the estimate of flow across the channel for the time interval we may obtain if we use the equation

$$F = R \times I$$

where

F = Total flow across the channel during time interval of length I

R = Channel capability, as already described

I = Length of time interval considered.

SCOPE already constrains planning increments in LIFTCAP to be at least as long as the longest cycle time, but such a restriction has proven insufficient for some problems.

1.2 Three-Phase Plan of Study

Georgia Tech will develop three strategies for resolving the current problem with the continuous flow assumption for sea assets; these will be a near-term solution (a "quick-fix"), a medium-term solution, and a long-term solution. The directions of study for these three phases of research are as follows:

(1) Near-term - development of a pre-processor for MRMATE (or, equivalently, a post-processor for LIFTCAP) to determine accurate delivery capability for each channel for each MRMATE planning period. This solution is presented later in this report.

(2) Medium-term - development of a post- and/or intermediate-processor for SCOPE to construct actual

routes for ships in such a way that the SCOPE solution is attained as closely as possible and actual delivery dates are obtained. This model should have the added benefit of resolving the problem of assignment of fractional ships to channels. Development of this model should be completed during fiscal 1987.

(3) Long-term - development of an optimization model with which to replace LIFTCAP for sea (although employment of air assets would still be planned by LIFTCAP). A review of technical literature suggests that very little work has been done on such problems, so an acceptable model here may lie several years in the future.

While the ideas for study listed above are probably the best candidates for each phase, they may not actually be those implemented; other ideas may be followed up also.

2.0 ANALYSIS OF THE SEVERITY OF THE CONTINUOUS FLOW ASSUMPTION

As previously mentioned, the accuracy of the continuous flow assumption depends on number of assets, transit times, cycle times, load (and un-load) times, and effective planning horizons. The following sections give graphical and numerical analysis (respectively) for how these factors affect model accuracy.

2.1 Graphical Analysis

Figures 1 through 24 give comparisons of SCOPE's solutions to "actual" solutions for individual channels having different

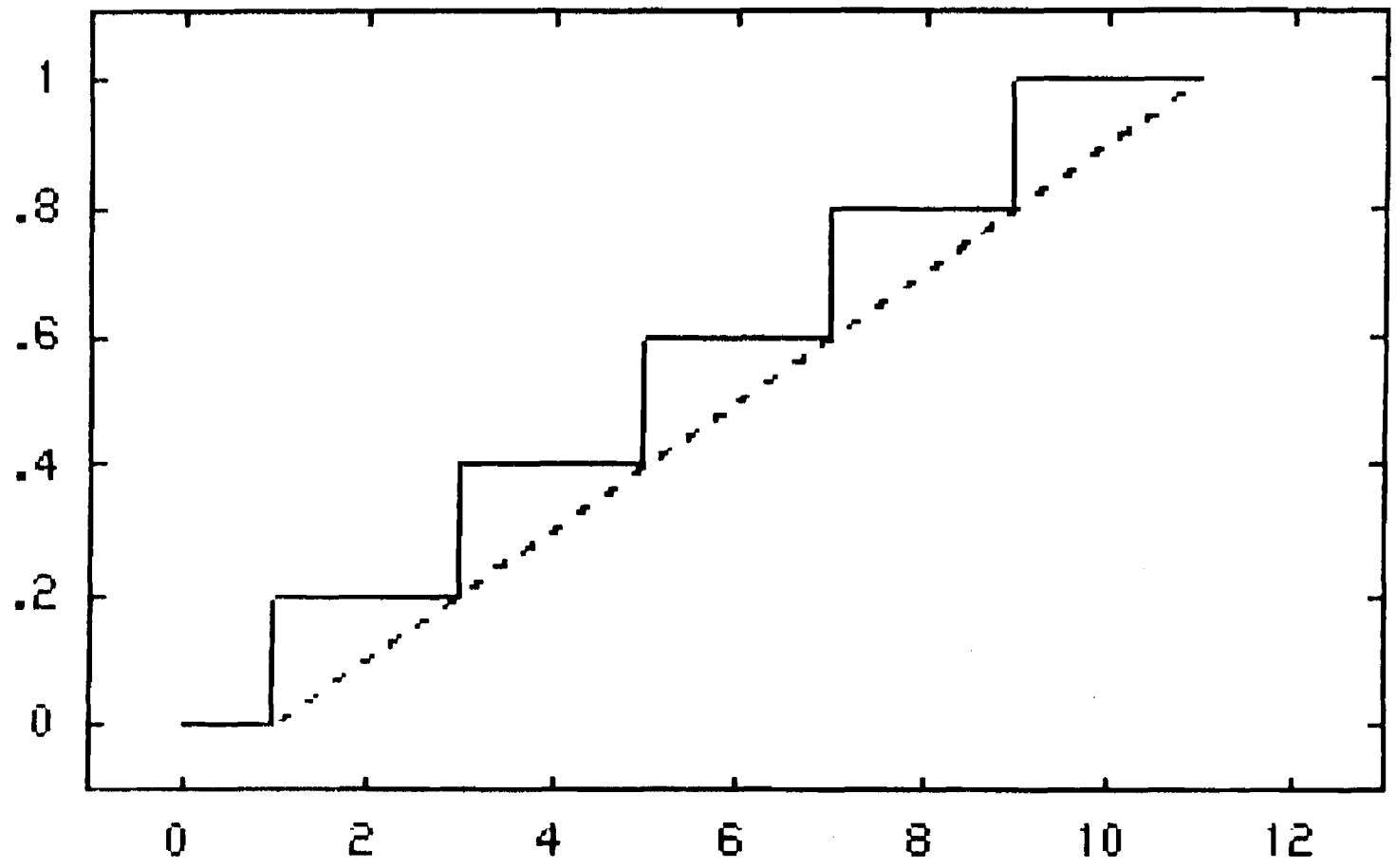


FIGURE 1. Sample Comparison of Cumulative Flow Over Time

Note: For the above, off-loading is considered instantaneous. The solid line denotes actual flow, and the broken line indicates the flow predicted by SCOPE for a channel.

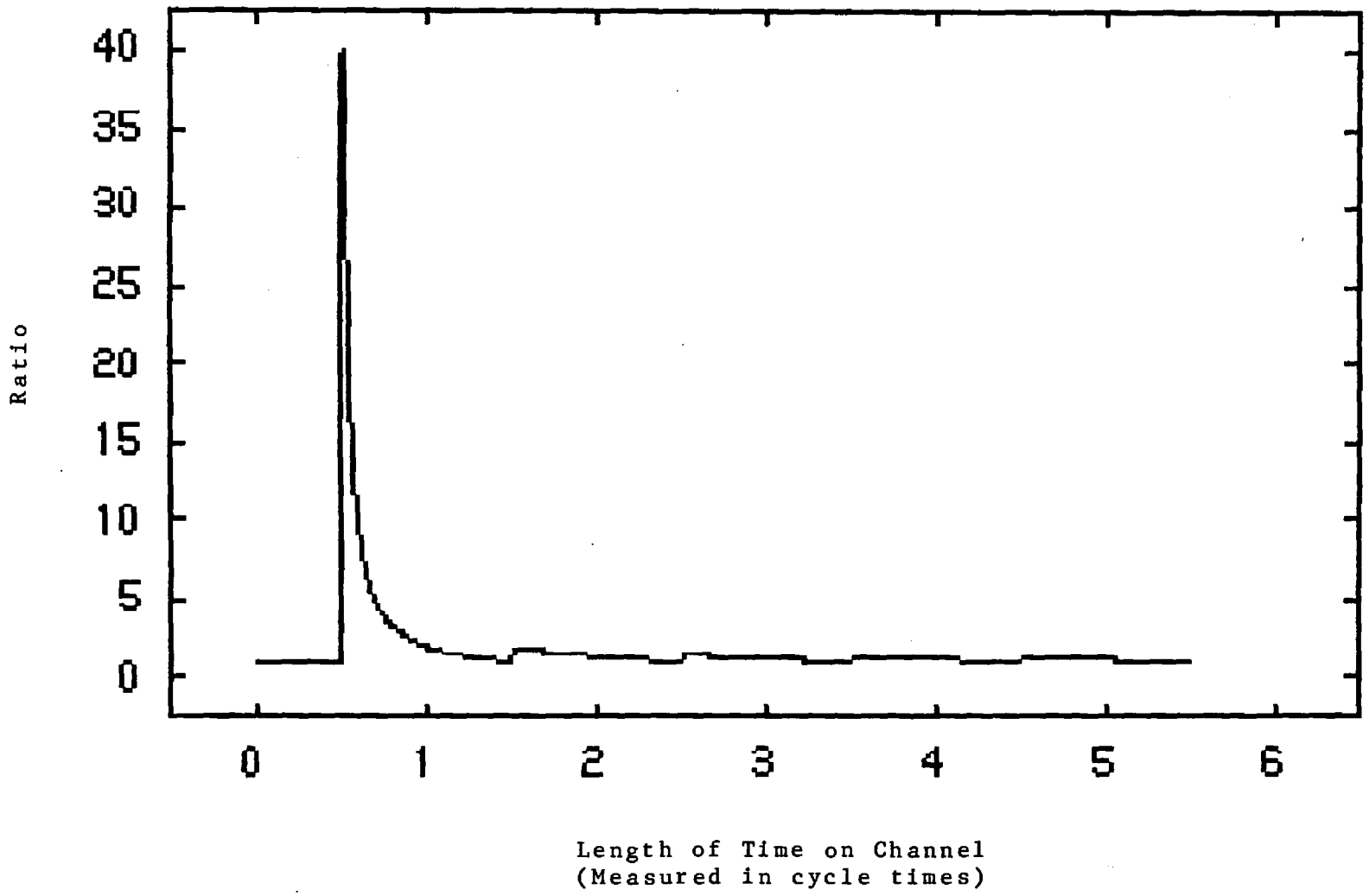


FIGURE 2. Ratio of Actual Flow to Reported Flow for One Ship (Low Resolution)

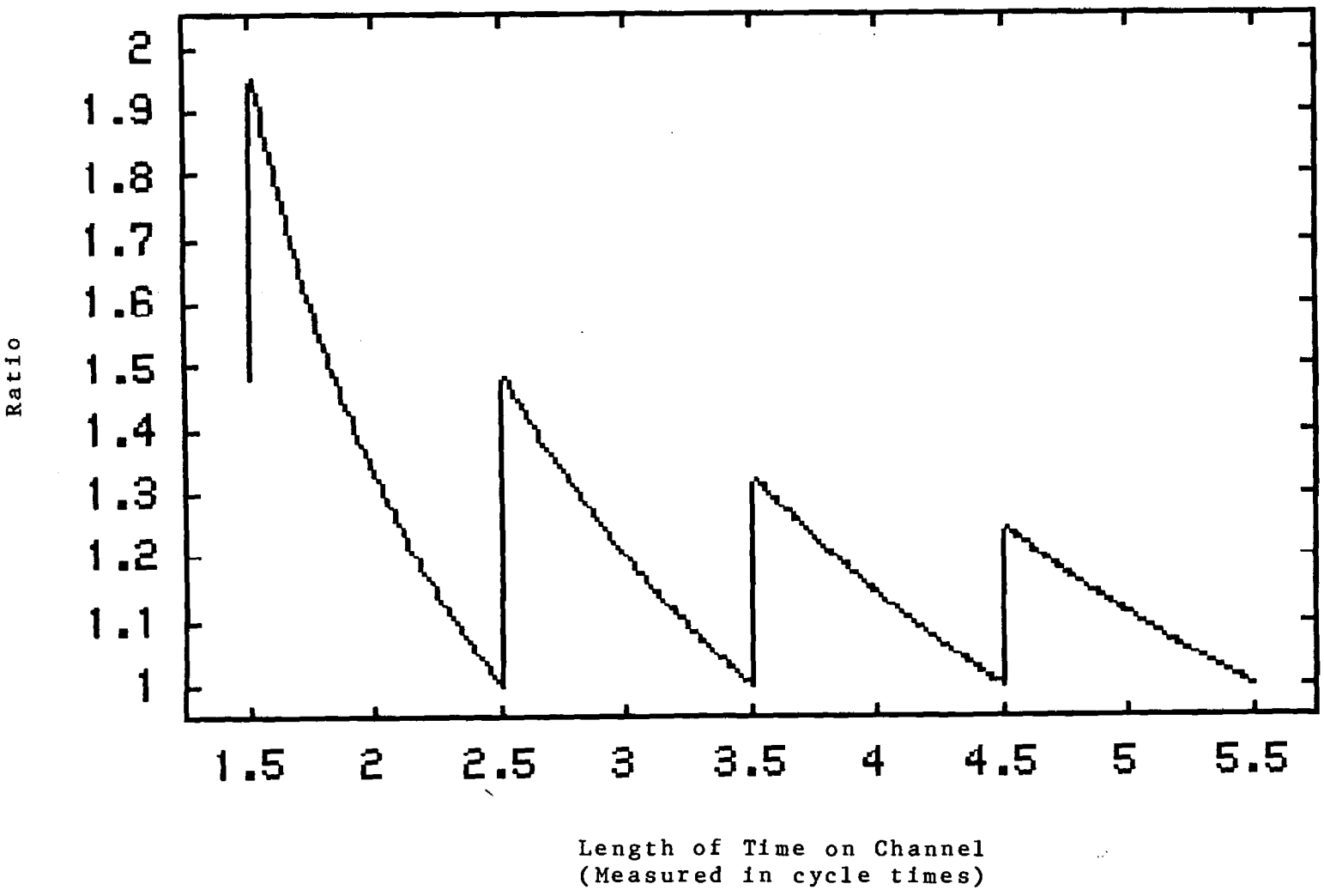


FIGURE 3. Ratio of Actual Flow to Reported Flow for One Ship With OL:CT = 2.5%

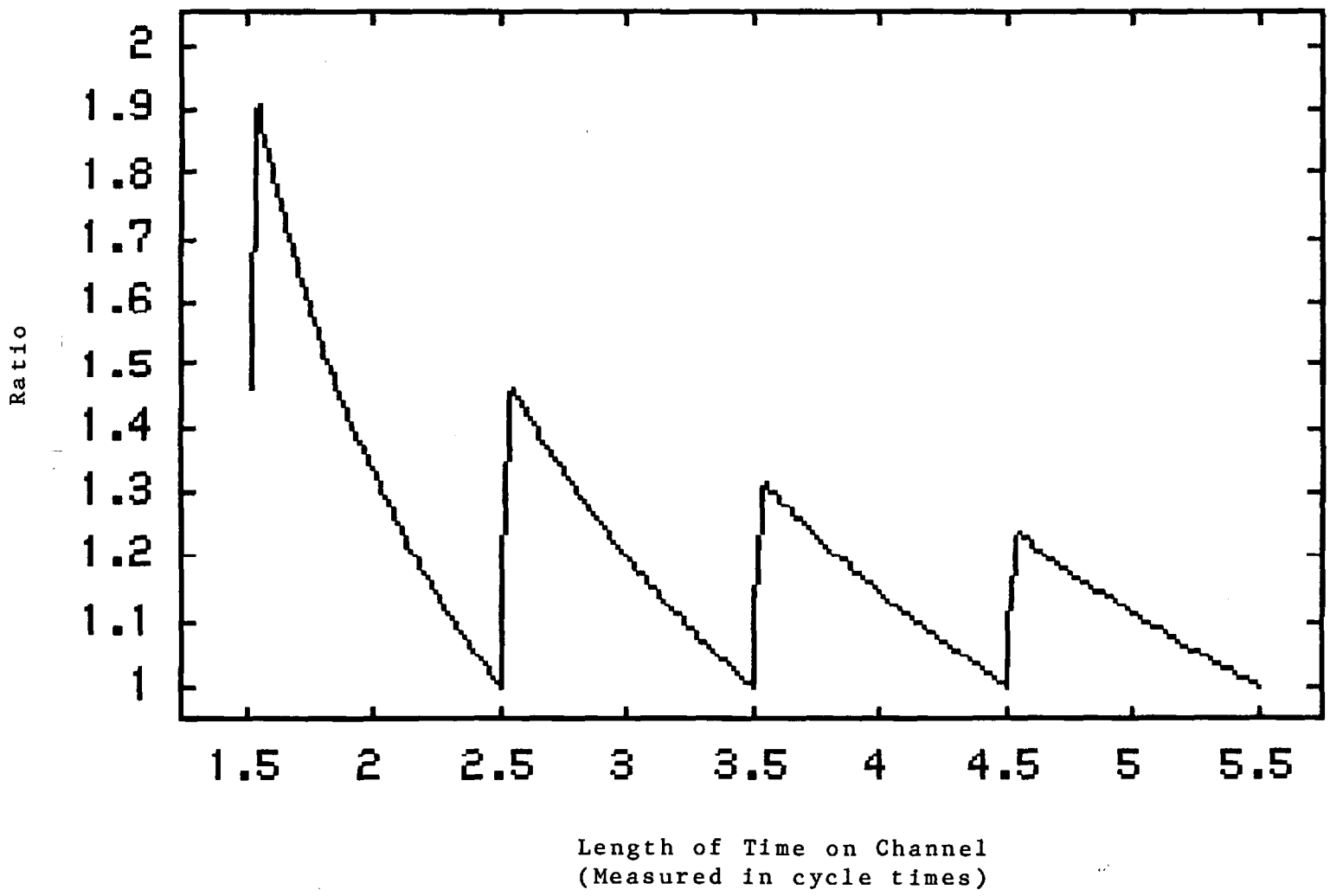


FIGURE 4. Ratio of Actual Flow to Reported Flow for One Ship With OL:CT = 5%

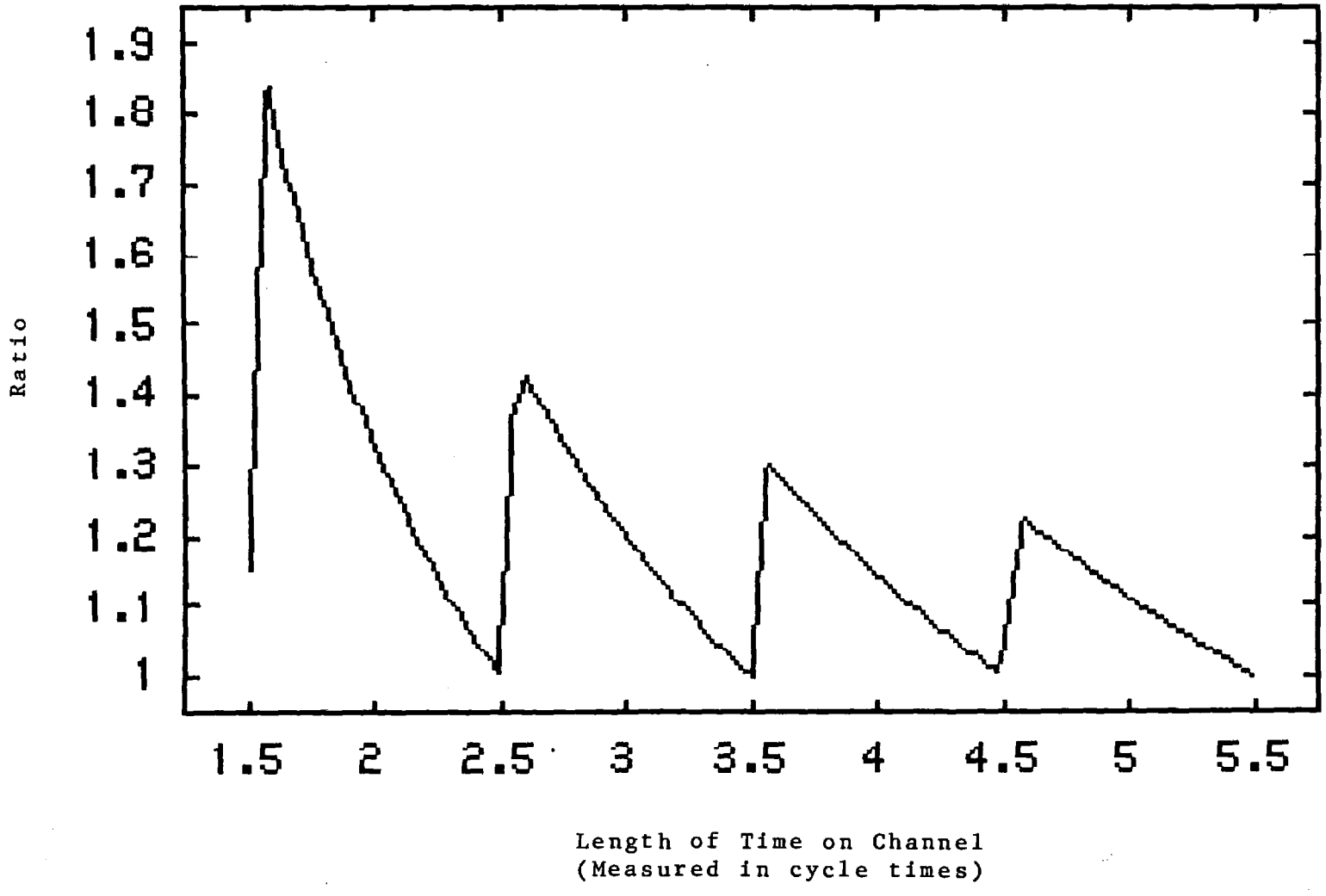


FIGURE 5. Ratio of Actual Flow to Reported Flow for One Ship With OL:CT = 7.5%

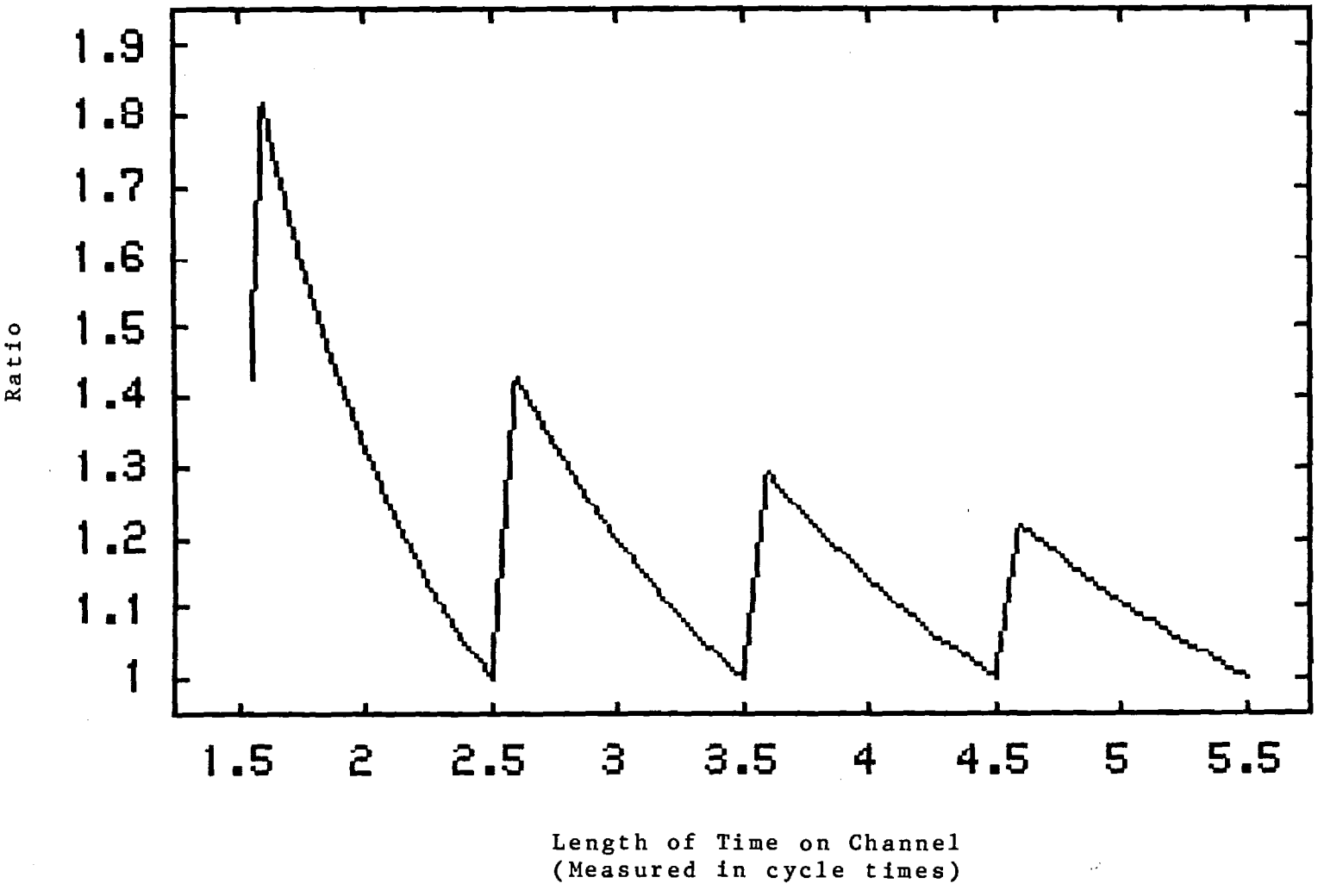


FIGURE 6. Ratio of Actual Flow to Reported Flow for One Ship With OL:CT = 10%

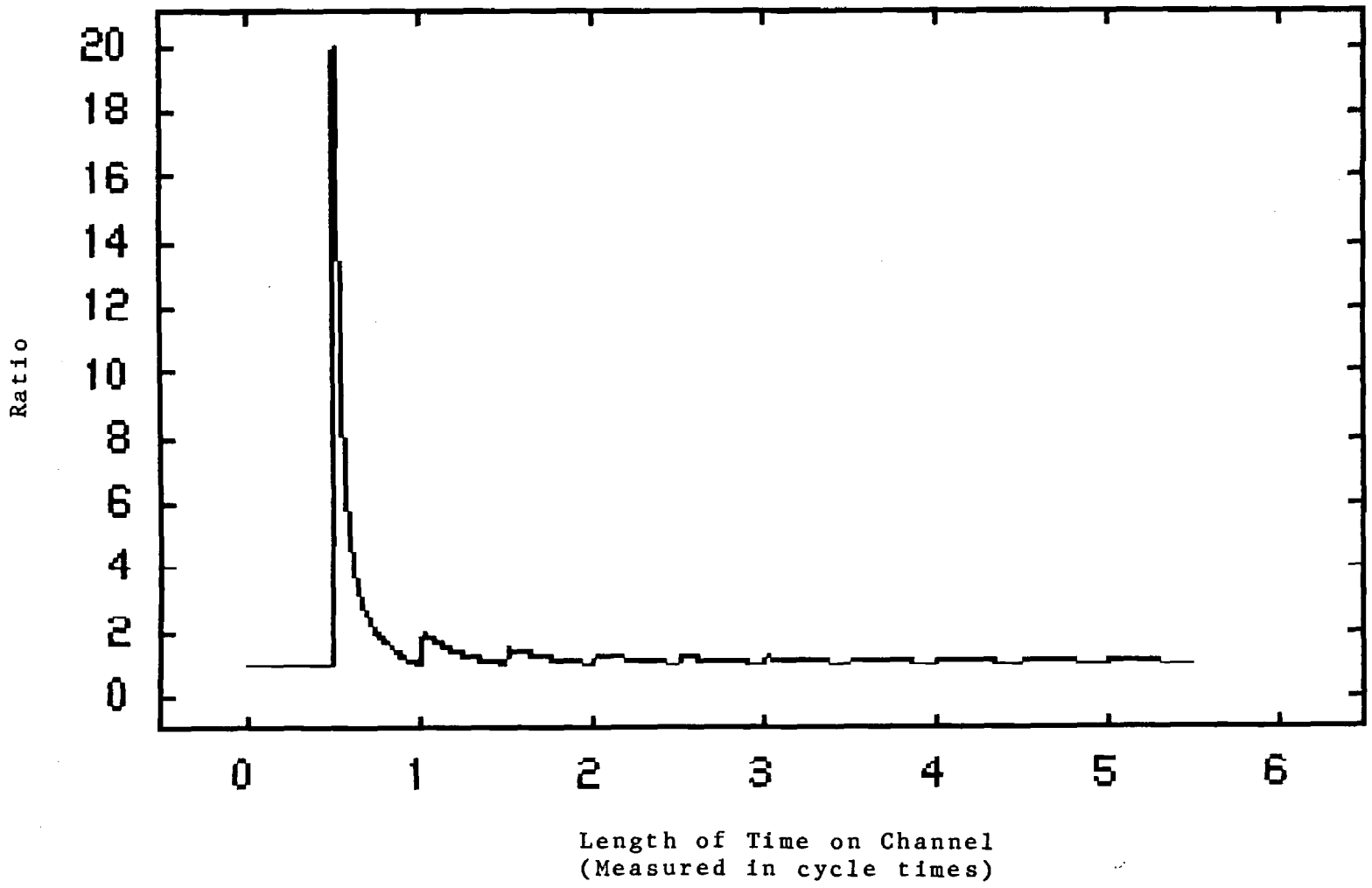


FIGURE 7. Ratio of Actual Flow to Reported Flow for Two Ships (Low Resolution).

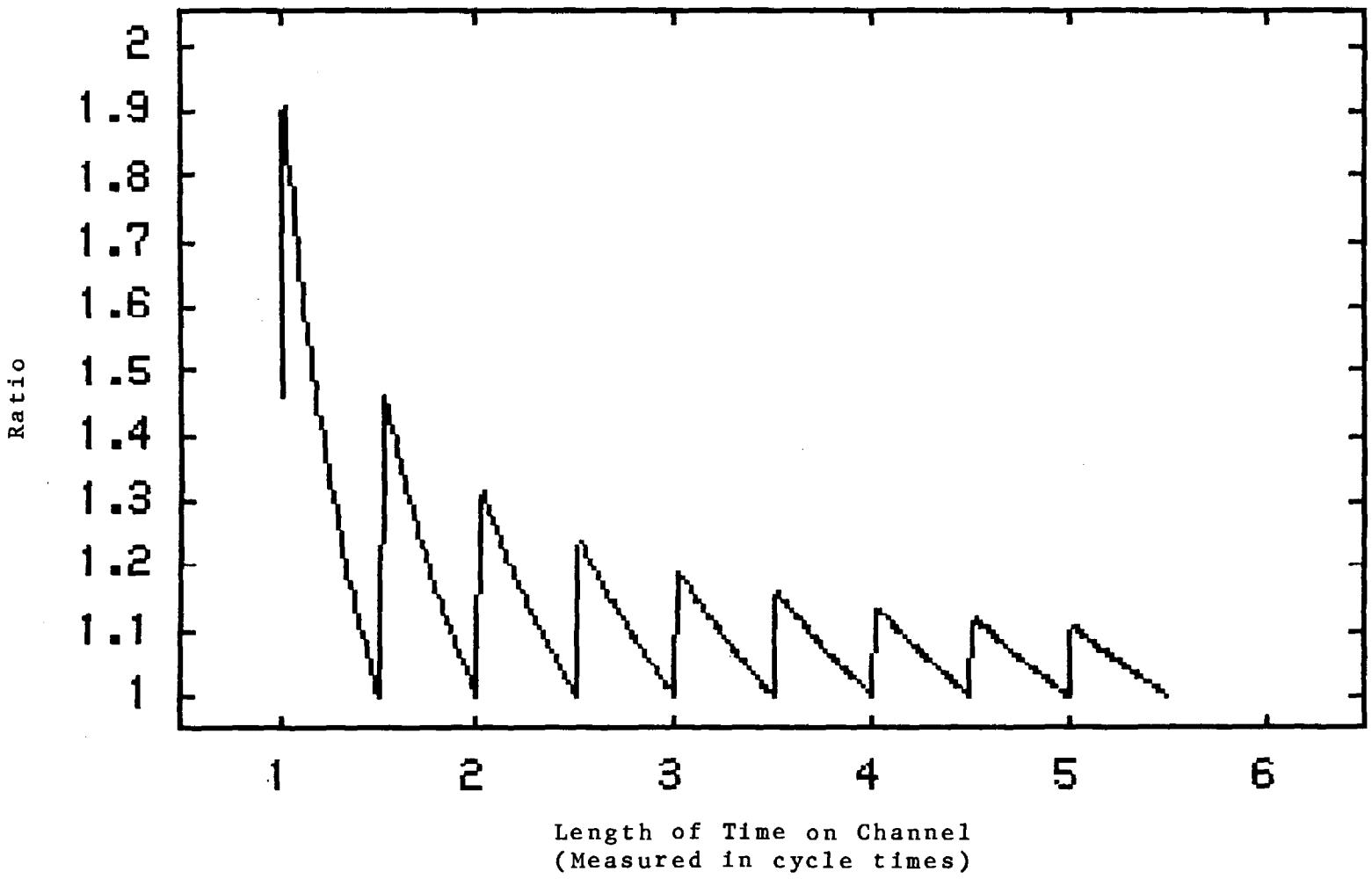


FIGURE 8. Ratio of Actual Flow to Reported Flow for Two Ships With OL:CT = 2.5%

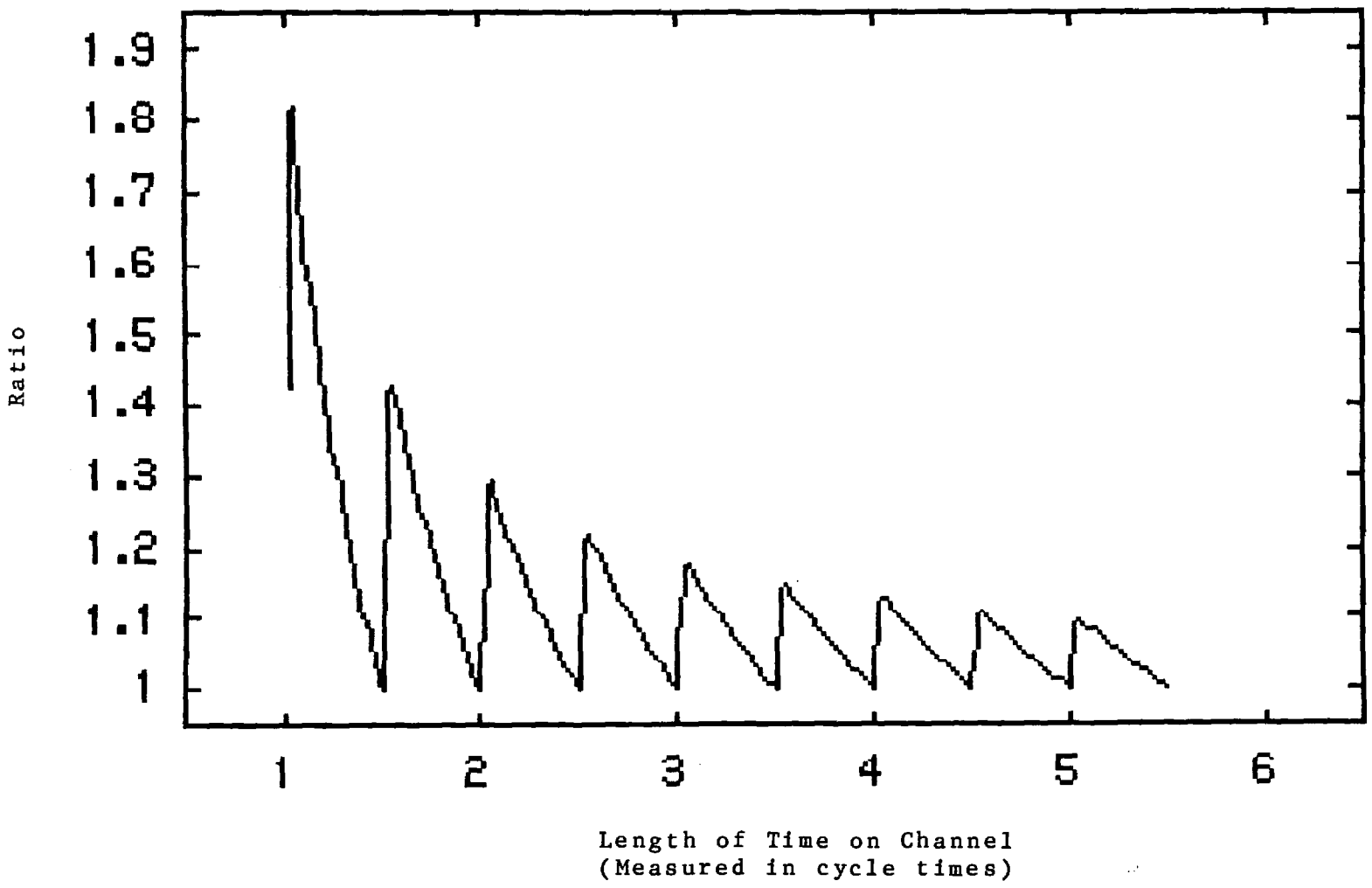


FIGURE 9. Ratio of Actual Flow to Reported Flow for Two Ships With OL:CT = 5%

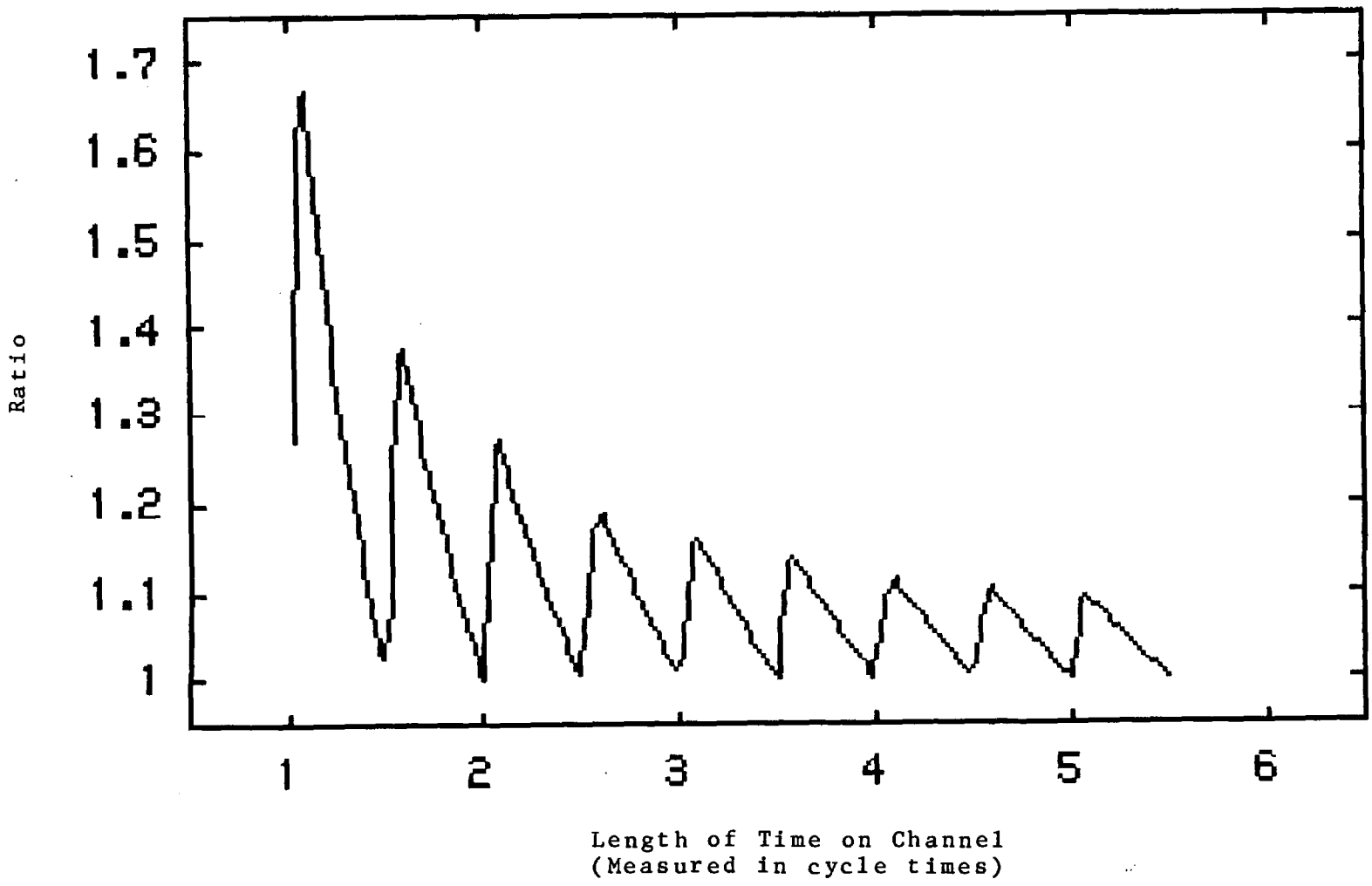


FIGURE 10. Ratio of Actual Flow to Reported Flow for Two Ships With OL:CT = 7.5%

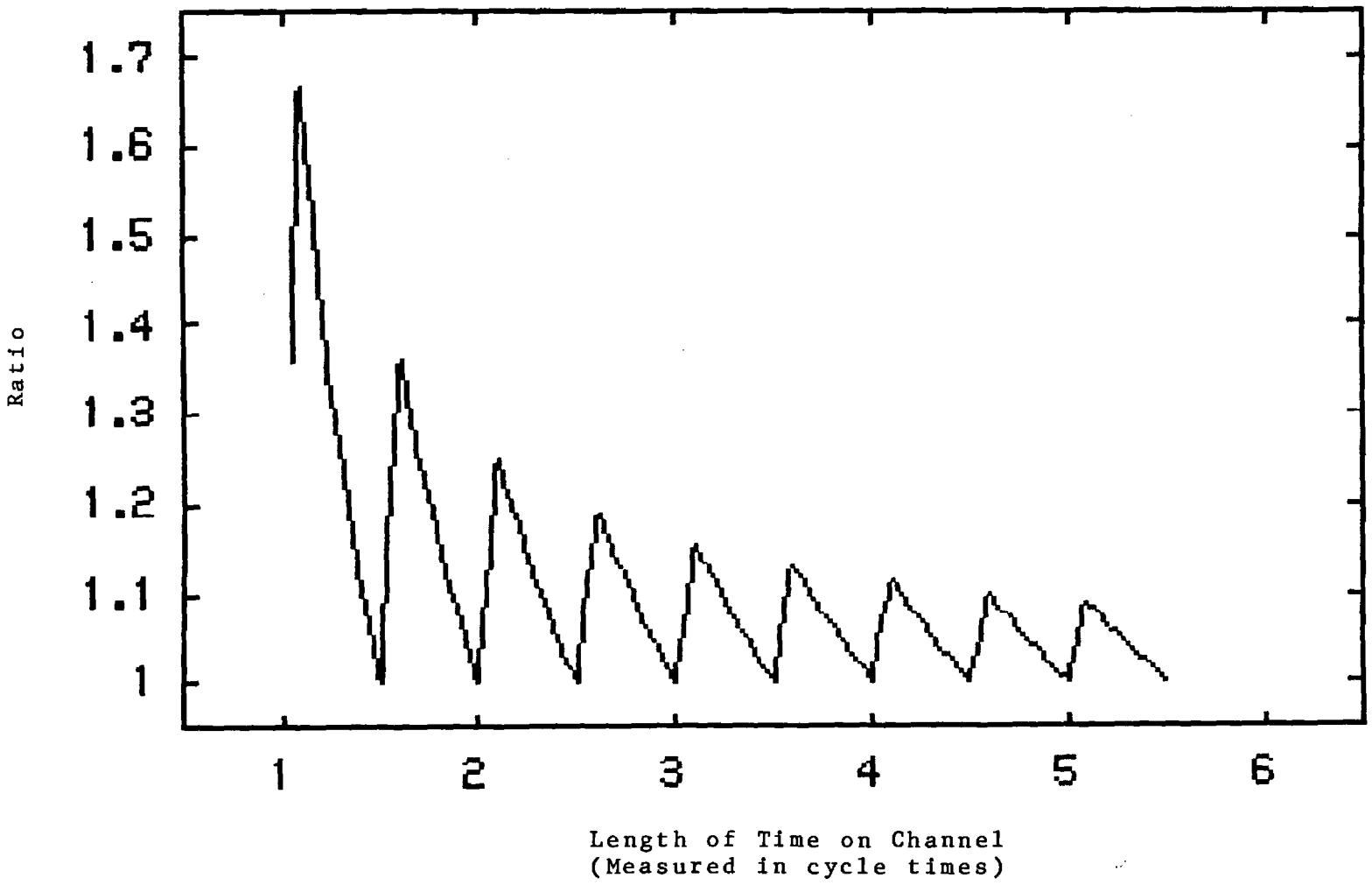


FIGURE 11. Ratio of Actual Flow to Reported Flow for Two Ships With OL:CT = 10%

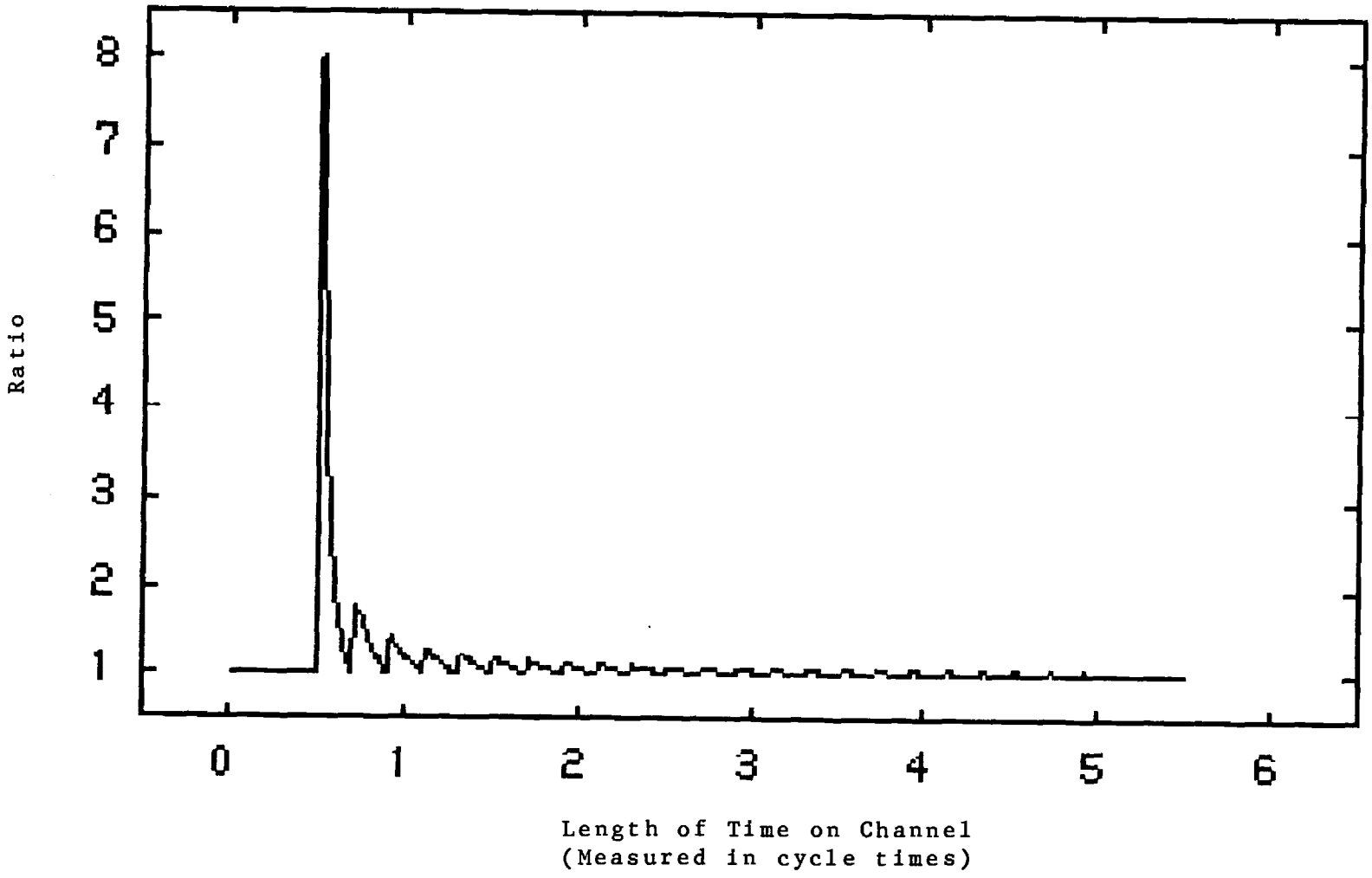


FIGURE 12. Ratio of Actual Flow to Reported Flow for Five Ships (Low Resolution)

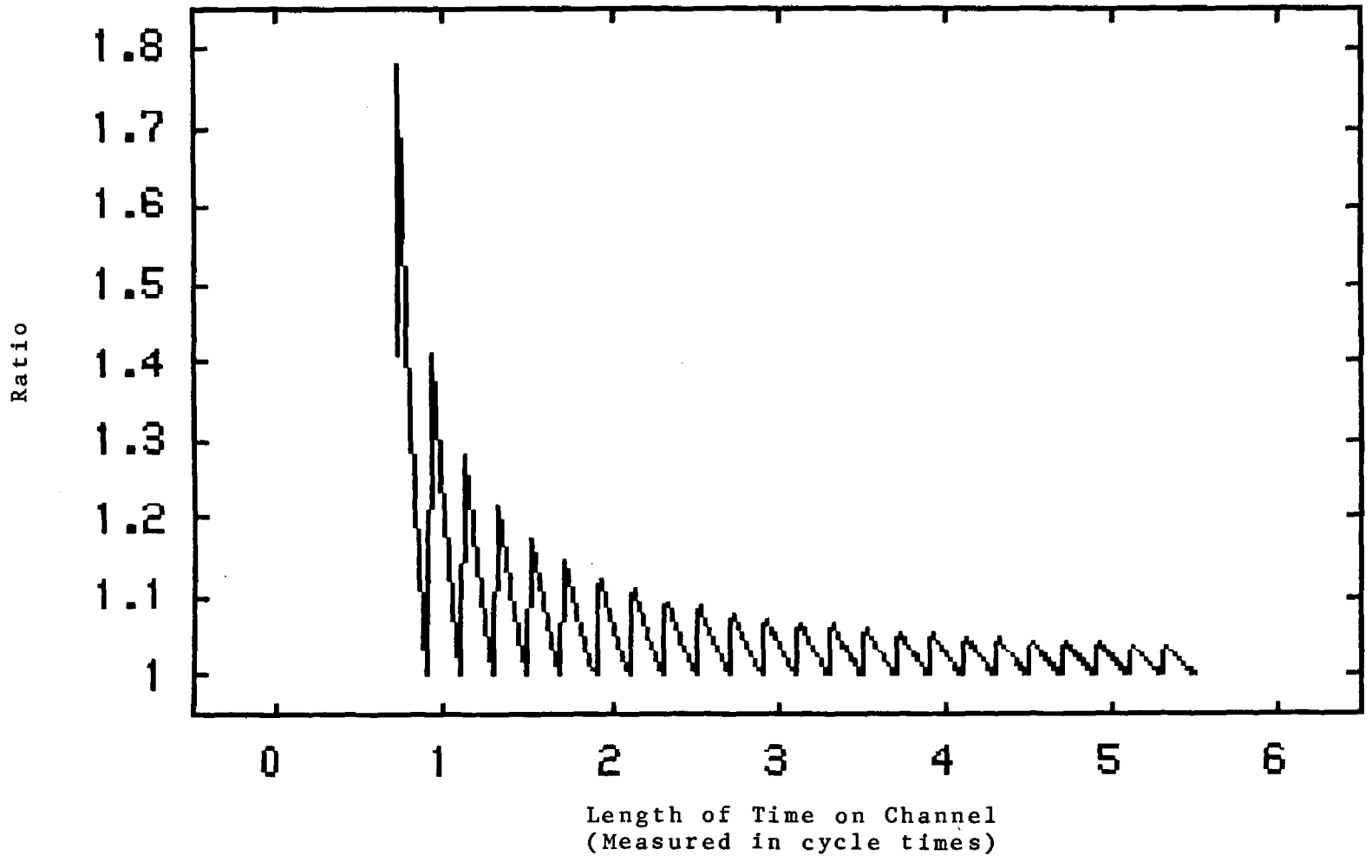


FIGURE 13. Ratio of Actual Flow to Reported Flow for Five Ships With OL:CT = 2.5%

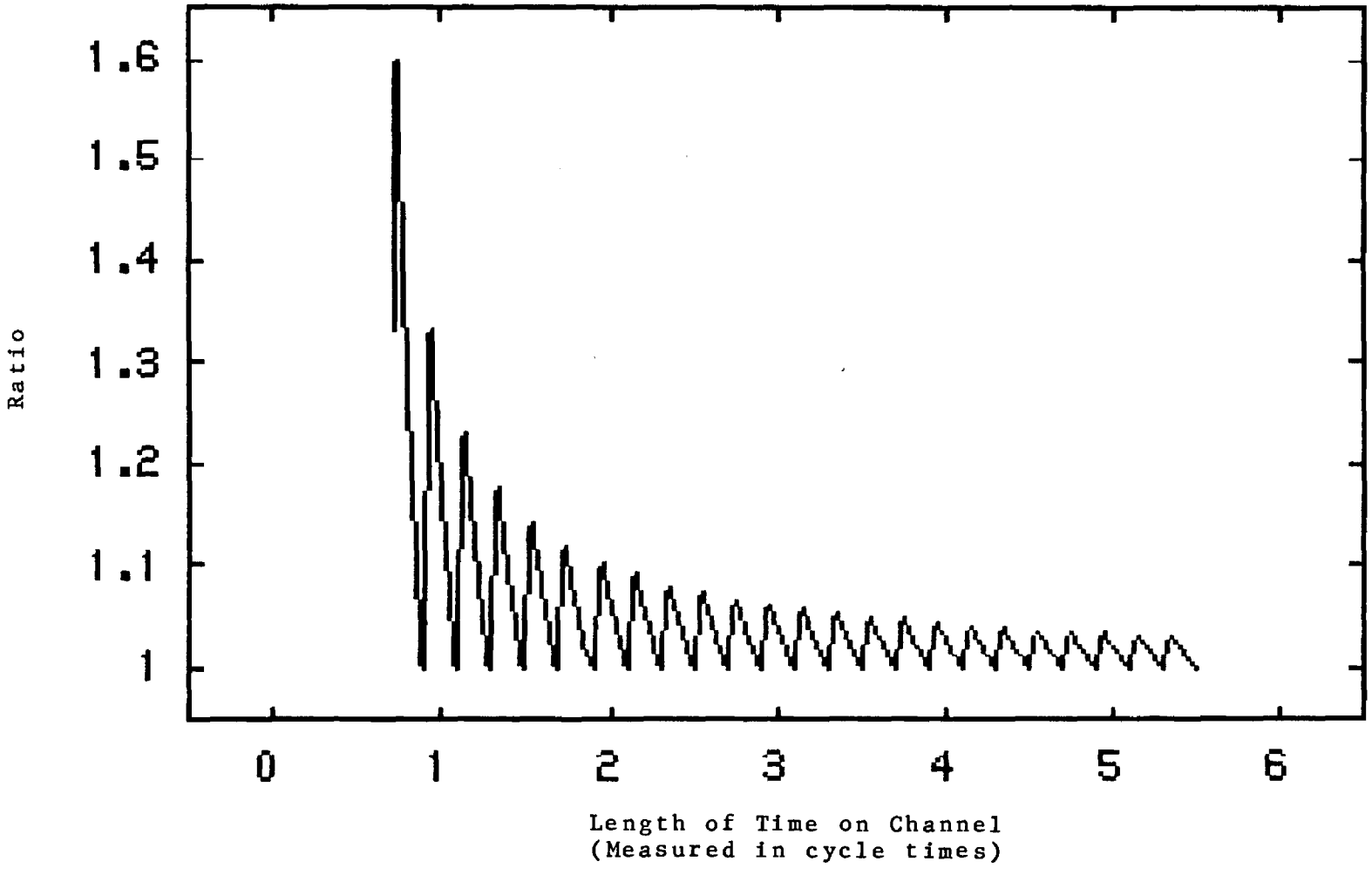


FIGURE 14. Ratio of Actual Flow to Reported Flow for Five Ships With OL:CT = 5%

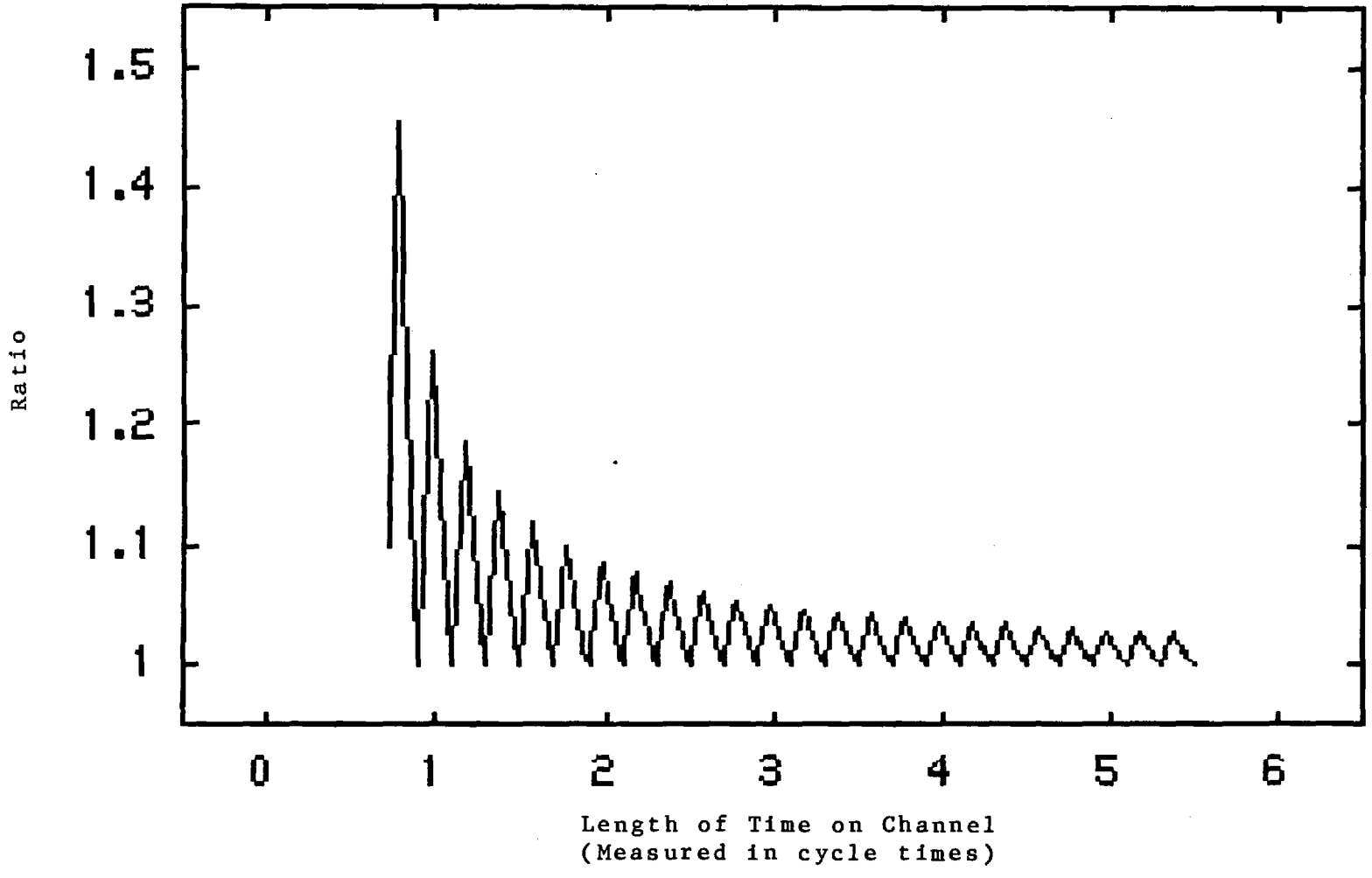


FIGURE 15. Ratio of Actual Flow to Reported Flow for Five Ships With OL:CT = 7.5%

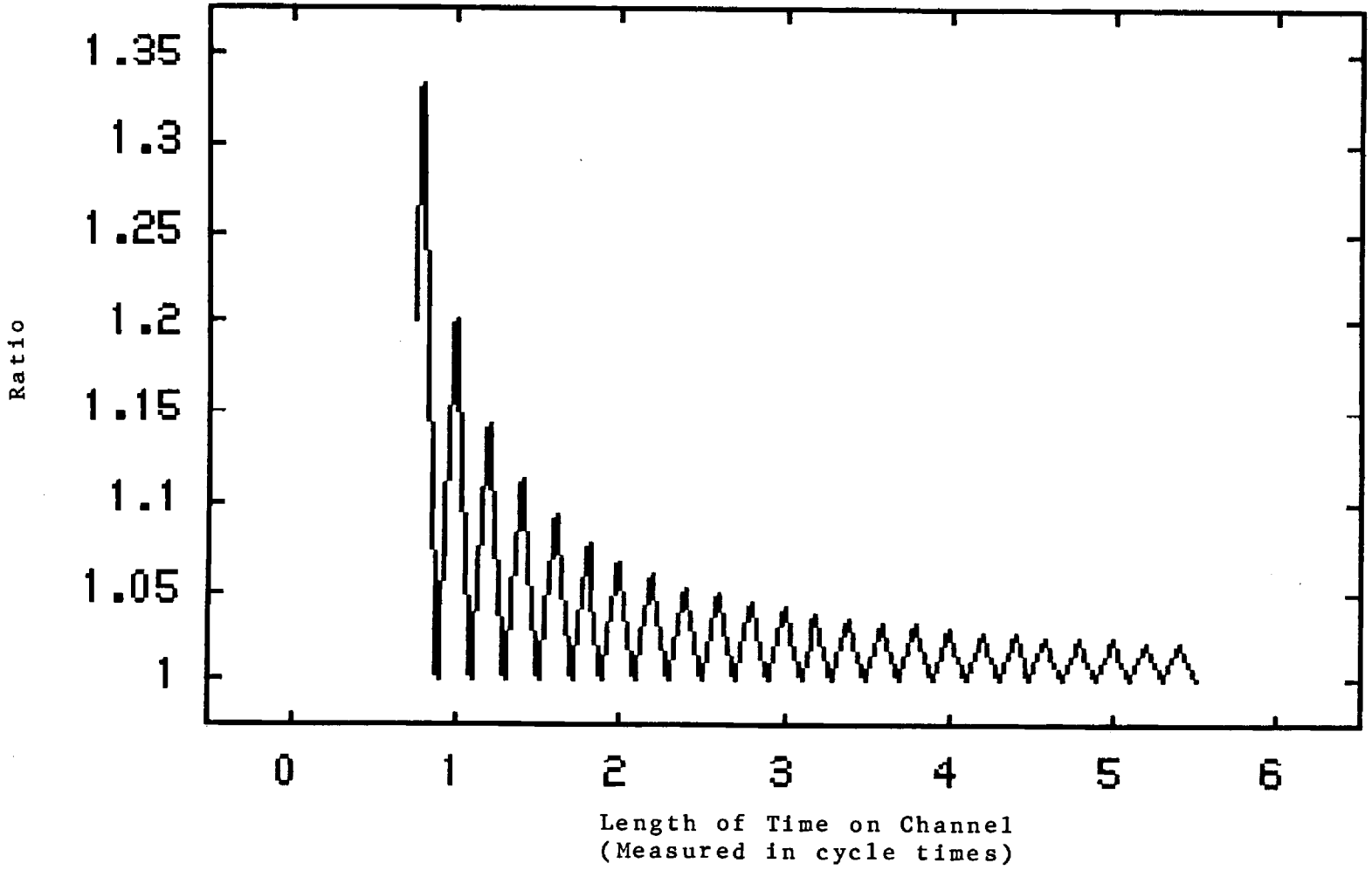


FIGURE 16. Ratio of Actual Flow to Reported Flow for Five Ships With OL:CT = 10%

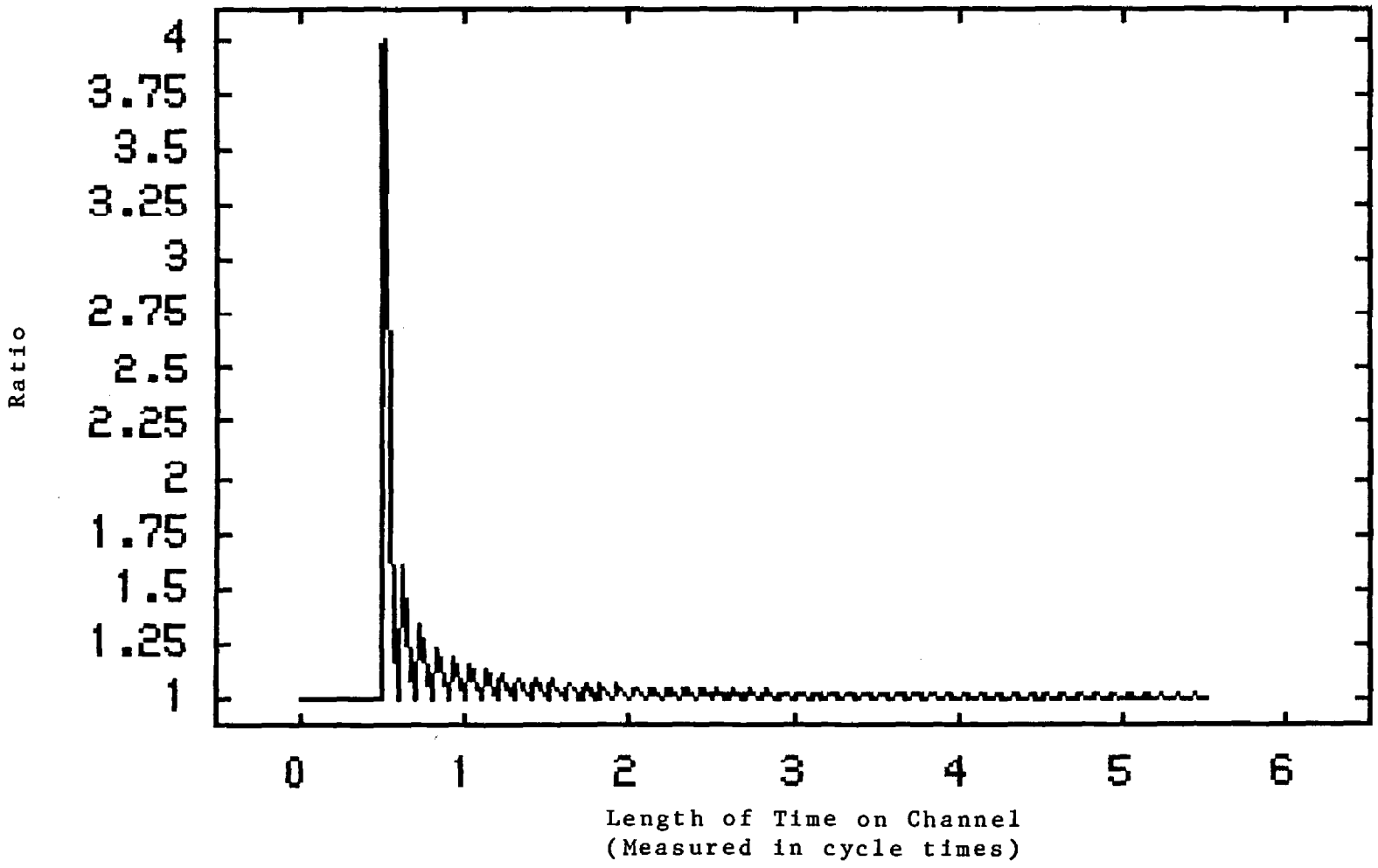


FIGURE 17. Ratio of Actual Flow to Reported Flow for Ten Ships With OL:CT = 2.5%

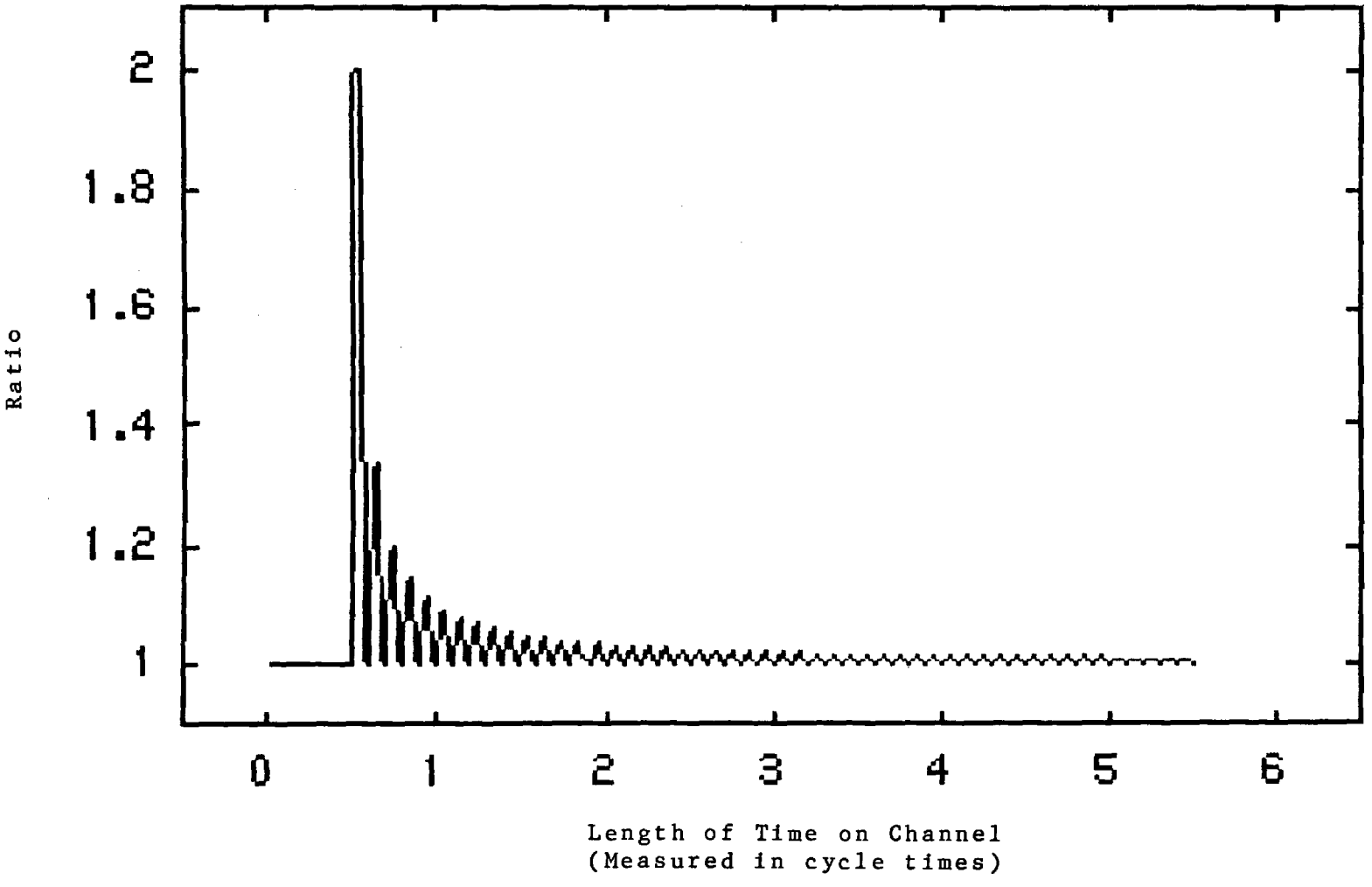


FIGURE 18. Ratio of Actual Flow to Reported Flow for Ten Ships With OL:CT = 5%

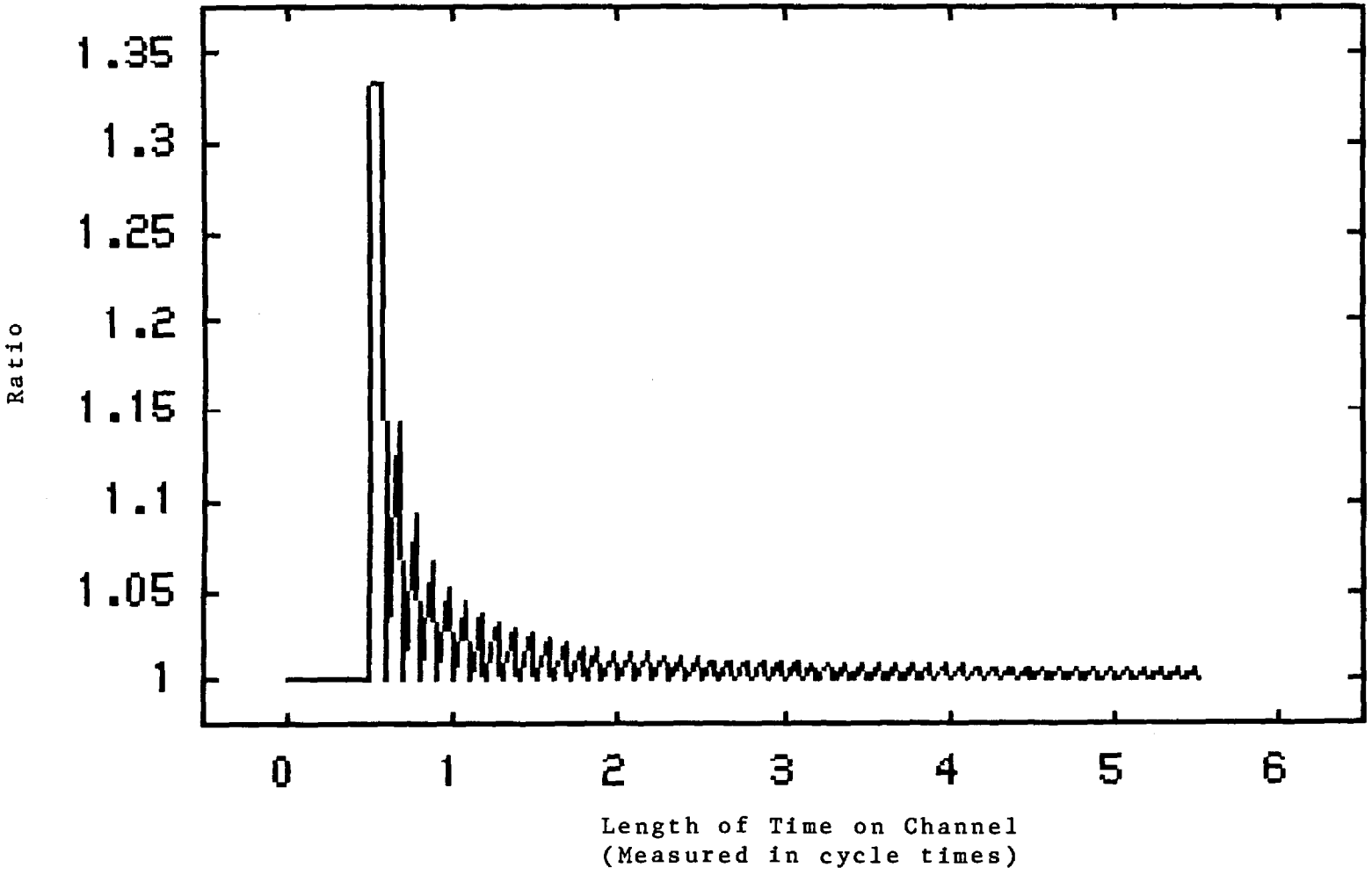


FIGURE 19. Ratio of Actual Flow to Reported Flow for Ten Ships With OL:CT = 7.5%

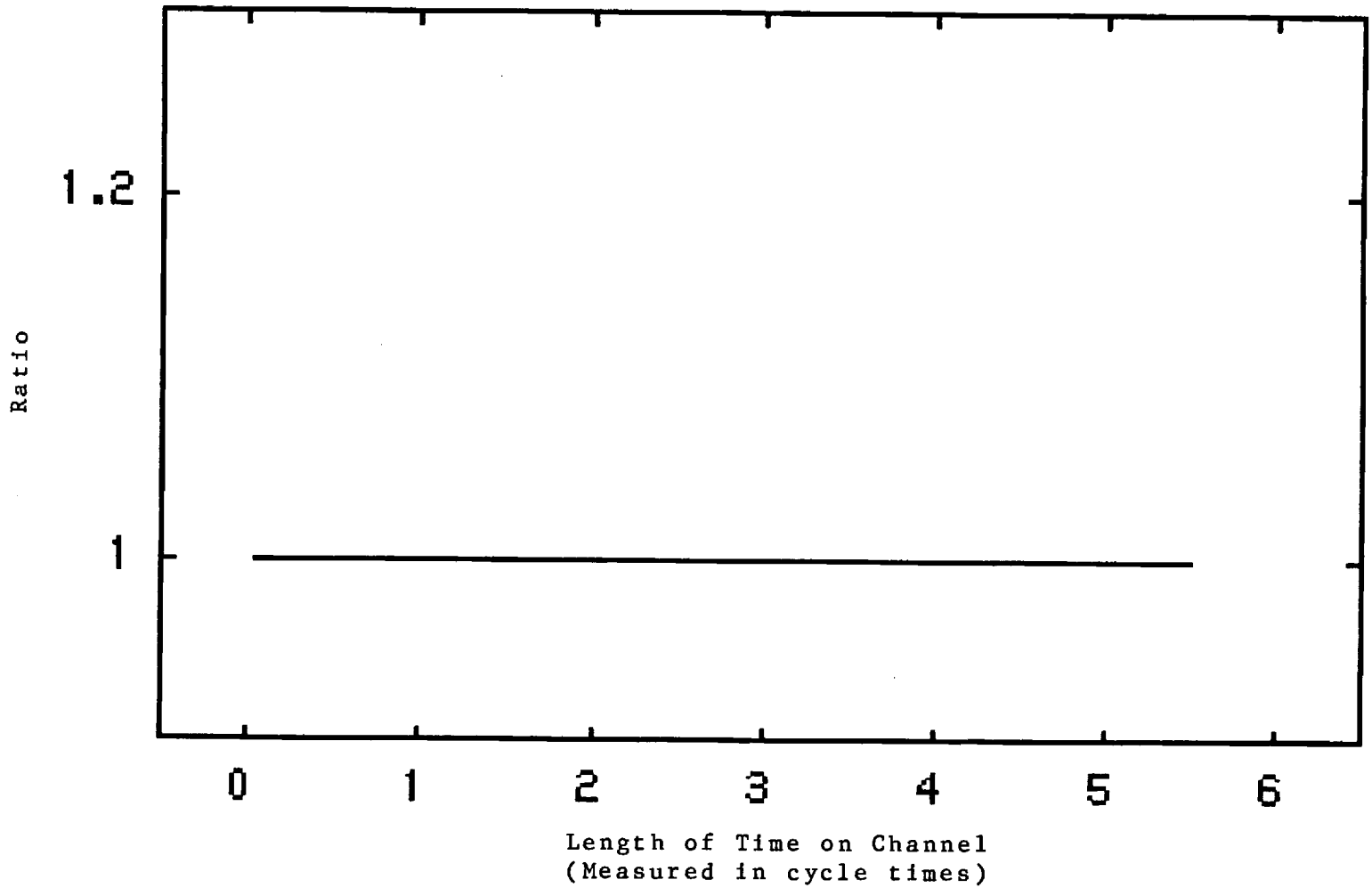


FIGURE 20. Ratio of Actual Flow to Reported Flow for Ten Ships With OL:CT = 10%

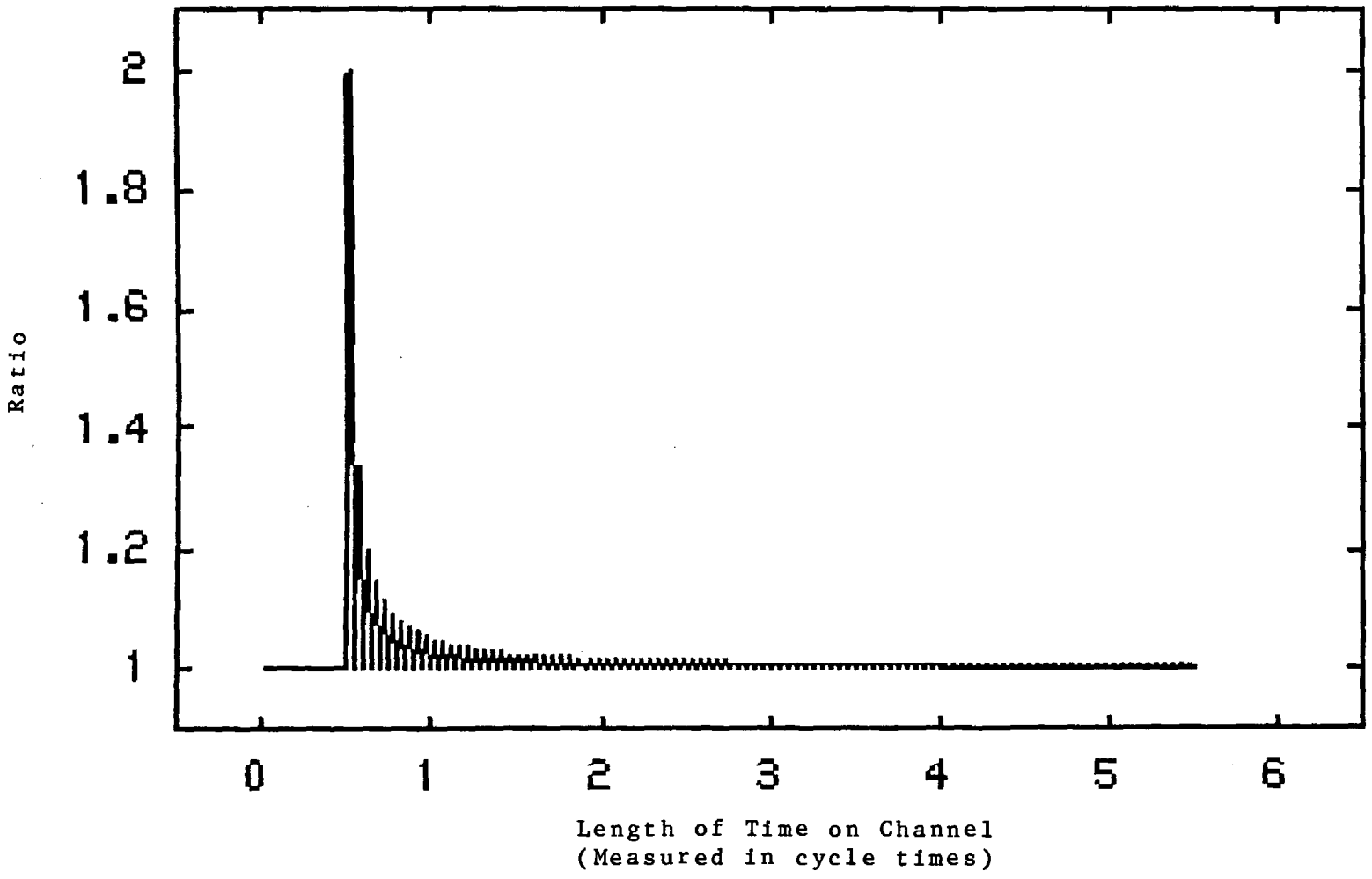


FIGURE 21. Ratio of Actual Flow to Reported Flow for Twenty Ships With OL:CT = 2.5%

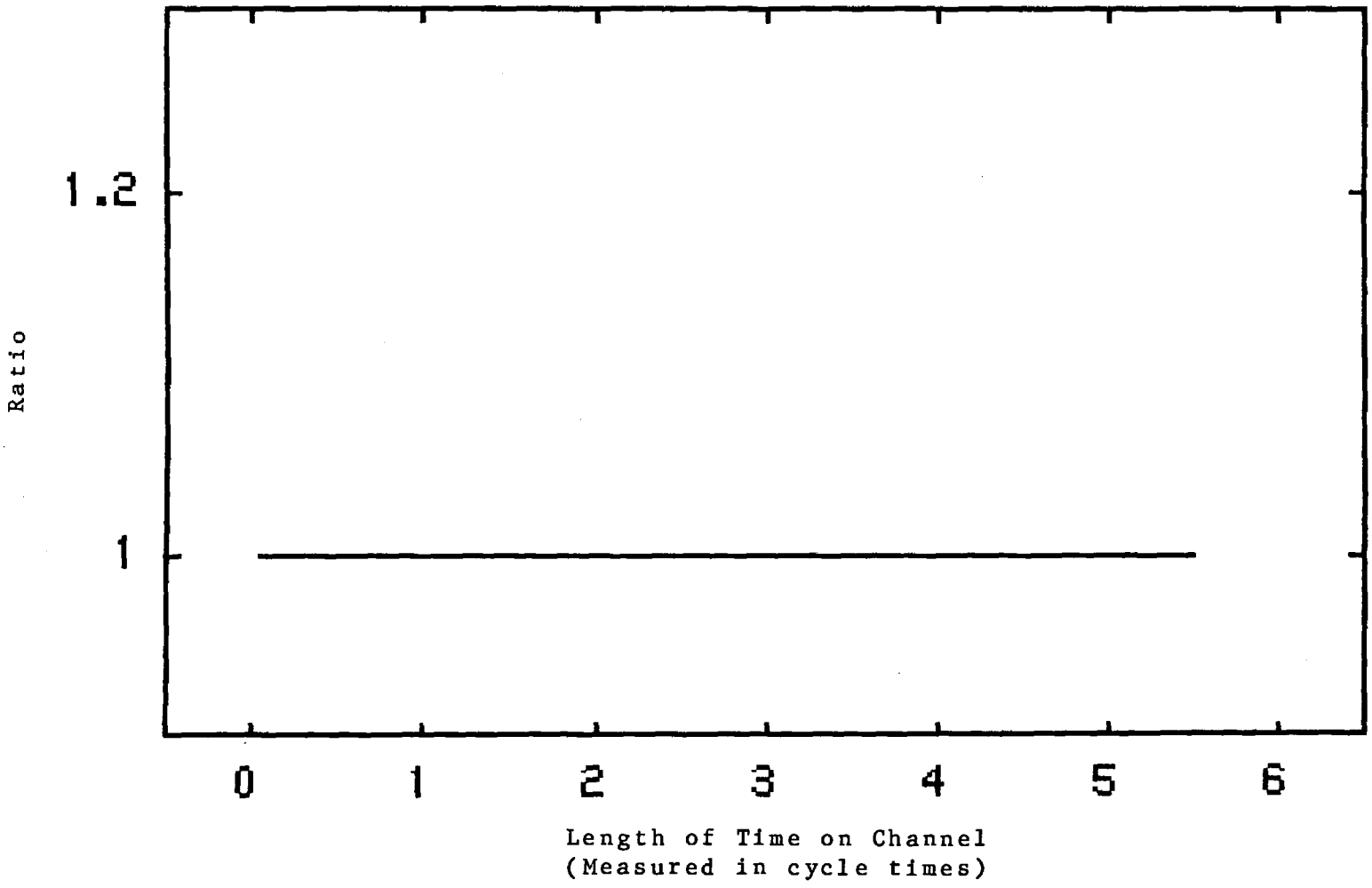


FIGURE 22. Ratio of Actual Flow to Reported Flow for Twenty Ships With OL:CT = 5%

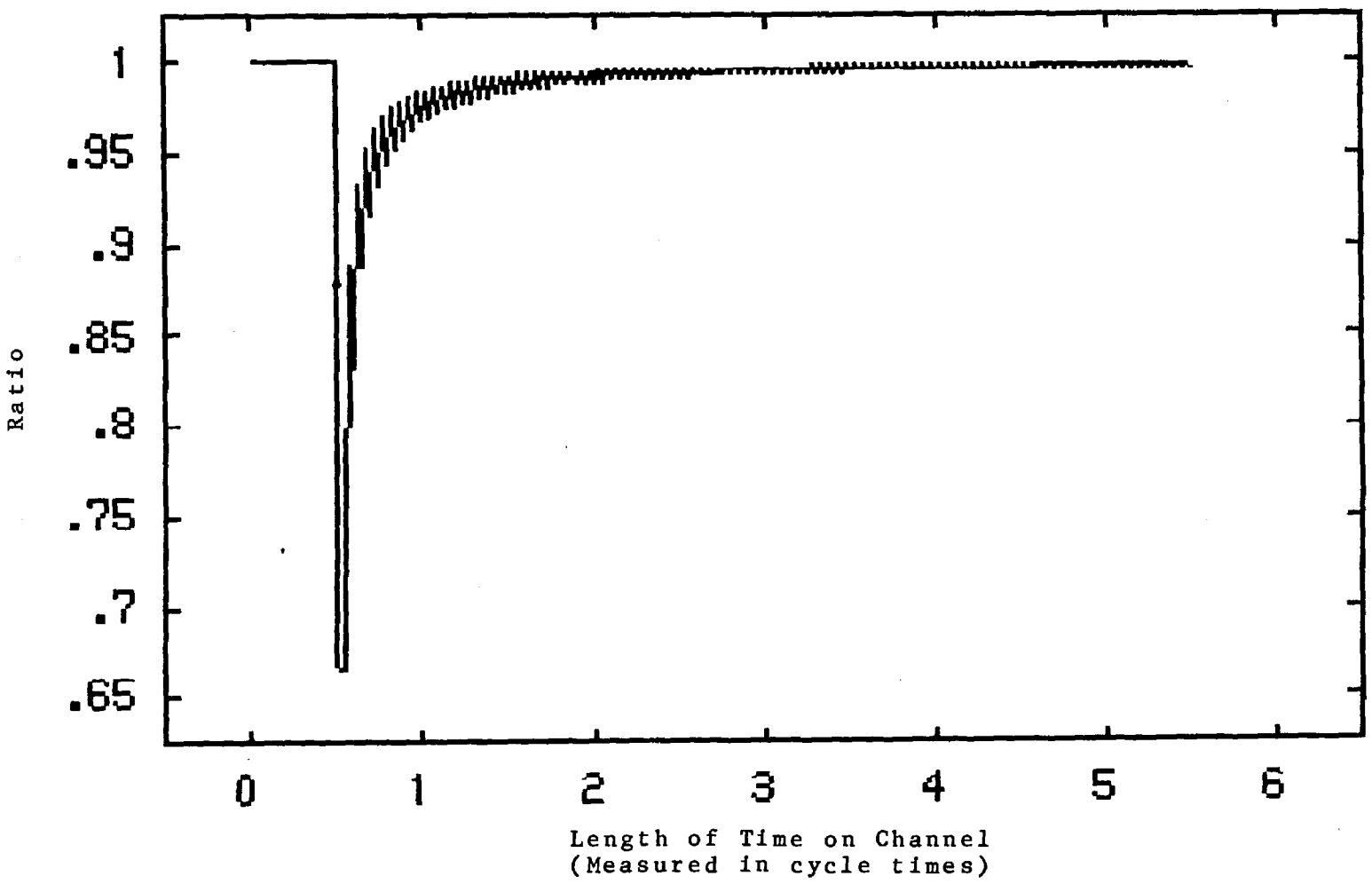


FIGURE 23. Ratio of Actual Flow to Reported Flow for Twenty Ships With OL:CT = 7.5%

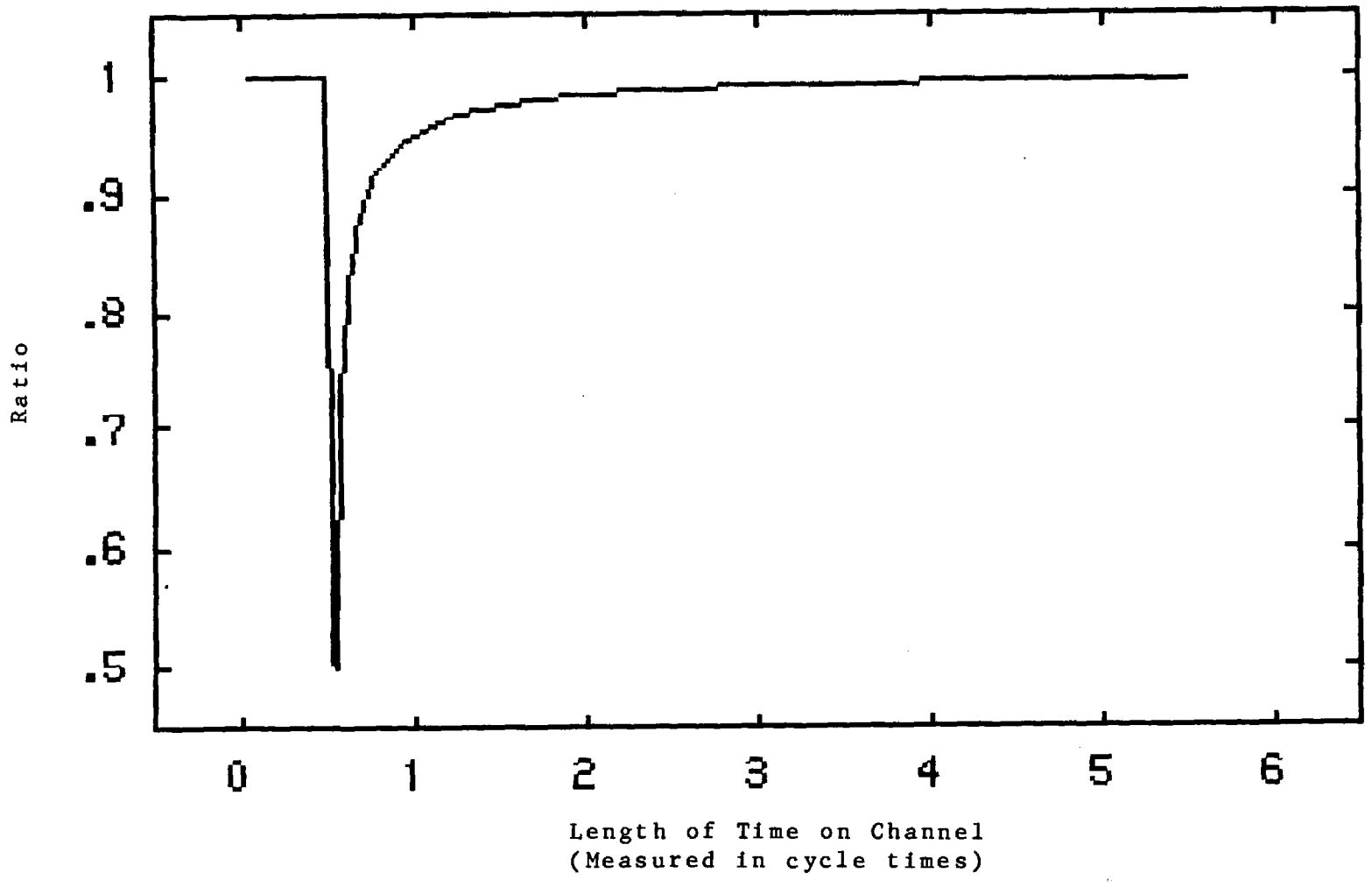


FIGURE 24. Ratio of Actual Flow to Reported Flow for Twenty Ships With OL:CT = 10%

characteristics. Figure 1 represents a channel with one ship and shows the fraction of flow occurring between time 0 (the start of the planning increment) and time 11 (11 transit times) for both discrete flow (actual, assuming instantaneous off-load time) and continuous flows (reported by SCOPE). Note that the SCOPE solution is correct on every odd transit time for this situation.

Figures 2 through 6 depict the ratio of actual flow to that reported by SCOPE over time for a channel with one ship. Figure 2 gives that ratio beginning at the start of the interval. Since the ratio is very large after one transit time, the resolution of the graph is low for most points. Figures 3 through 6 gain resolution by ignoring some of the initial observations, and depict this same ratio for ratios of off-load time to cycle time (OL:CT) of 2.5%, 5%, 7.5%, and 10% respectively.

Figures 7 through 24 are based on an assumption that ships will be spaced evenly on the channel. (For example, if a channel has 3 ships and the cycle time is 30 days, a ship would depart POE every 10 days.) Figures 7 through 11 are for a two-ship channel, Figures 12 through 16 are for a five-ship channel, Figures 17 through 20 are for a 10-ship channel, and Figures 21 through 24 are for a twenty-ship channel.

Note that when the number of ships on a channel is less than 1 / (OL:CT), SCOPE initially underestimates lift capability but converges to accuracy as the time interval lengthens. When the number of ships on a channel is greater that 1 / (OL:CT), SCOPE initially overestimates lift capability and then converges to accuracy over time. If the number of ships is equal to 1 /

(OL:CT), the SCOPE solution is accurate. Of course, this analysis depends on the assumption of how ships are used. Equal spacing on a channel may not actually be the best way to use ships in all cases, but it does seem to be the best assumption to make.

2.2 Numerical Analysis

The foregoing graphical analysis is based on the numerical analysis presented in this section. A description of flow calculation using the continuous flow assumption (as currently performed by SCOPE) is presented first, and is followed by explanation of flow calculation for discrete ships restricted by the equal spacing assumption previously described.

2.2.1 Flow Calculation Using Continuous Flow Assumption

Under the continuous flow assumption, the cumulative flow across a channel at time T, denoted by FC(T), is found according to

$$FC(T) = (N/CT) \times (T-TO-D) \times A, \text{ if } T-TO > D$$
$$= 0 \quad , \text{otherwise}$$

where

N = Number of ships assigned to channel

CT = Cycle time on channel

T = Time at which cumulative flow is to be determined

TO = Time at which planning increment begins

D = Initial delay, or length of time after TO during which no ship can arrive at POD

A = Capacity per ship.

For illustration, consider the following example:

N = 5 , CT = 12 days, TO = 0, D = 6 days,

A = 10,000 stons. Find FC(21).

Using the above equation, we find that

$$FC(21) = (5/12) \times (21-6) \times 10{,}000$$

$$= 62{,}500 \text{ stons.}$$

2.2.2 Flow Calculation for Discrete Ships (Equal Spacing)

For this calculation, first define the following:

Y(T) = Number of ships that have arrived at POD up to time T.

U(T) = Number of ships that have completely off-loaded at POD up to time T

OL = Time required to unload a ship

TY(T) = Time at which last ship arrived at POD up to time T.

F(T) = Cumulative flow across channel completed by time T using discrete ships.

F(T) can then be found according to the following equations:

$$Y(T) = \text{INT}((N/CT) \times (T-TO-D)) + 1, \text{ if } T-TO \geq D$$
$$= 0 \text{ , otherwise}$$

$$U(T) = \text{INT}((N/CT) \times (T-TO-D-OL)) + 1, \text{ if } T-TO-OL \geq D$$
$$= 0 \text{ , otherwise}$$

$$TY(T) = D + \frac{(Y(T)-1) \times CT}{N}, \text{ if } Y(T) > 0$$
$$= \text{undefined} \text{ , otherwise}$$

$$F(T) = A \times \left((U(T) + \sum_{i=1}^{Y(T)-U(T)} \frac{((i-1)\times(CT/N))+T-TY(T))}{OL}\right) ,$$
$$\text{if } Y(T)-U(T) > 0$$
$$= A \times U(T) \text{ , otherwise.}$$

Some remarks on this formulation are useful. First, if the

number of ships assigned to a channel is non-integral, integral ships are still used; only the spacing is changed to achieve the same average flow. Second, note that $Y(T) - U(T) \leq INT((N/CT) \times OL) + 1$, i.e., the maximum number of ships at POD at any point in time is found by rounding up the off-load time multiplied by the inverse of the channel spacing, thus providing one way of checking violation of port capacity constraints.

For illustration, let us return to the example used for flow calculation under the continuous flow assumption:

N = 5, CT = 12 days, TO = 0, D = 6 days,

A = 10,000 stons, OL = 2 days. Find F(21).

Using the above equations, we find that

$$Y(21) = INT((5/12) \times (21-6)) + 1$$
$$= INT(6.25) + 1$$
$$= 7.$$

$$U(21) = INT((5/12) \times (21-6-2)) + 1$$
$$= INT(5.42) + 1$$
$$= 6.$$

$$TY(21) = 6 + \frac{(7-1) \times 12}{5}$$
$$= 20.4.$$

$$F(21) = 10{,}000 \times (6 + (21-20.4)/2)$$
$$= 63{,}000 \text{ stons.}$$

So, the difference between the two answers for this case is 500 stons; the SCOPE solution has 0.79% error.

3.0 NEAR-TERM SOLUTION

The recommended near-term modification to SCOPE consists of

a pre-processor to MRMATE which will determine accurate lift capabilities for each sea node in MRMATE using both the capability rates provided by LIFTCAP and the asset and channel data provided as input to the model. This solution method will not guarantee that the closure date reported to the user will be accurate to the day, since the accuracy of the actual SCOPE optimization model is only to the MRMATE time period; it is the disaggregation post-processor (to SCOPE) that purports to provide actual closure estimates in days, so greater accuracy in closure estimates can be gained only by implementation of a two- or three-phase approach (as has been recommended by Georgia Tech in the past) or by modification of the disaggregation heuristic. The near-term modification may still be thwarted by those intent on doing so.

3.1 Algorithm for MRMATE Pre-Processor

The information needed by the pre-processor consists of the lift capability (rate) for each open channel for each LIFTCAP planning increment, the cycle time, transit time, additional initial delay, off-load time, and asset type capacity for each asset/channel pair. It has been discovered that much of this information is not kept very accurately, so it is important to remember that the model can be no more accurate than the data.

Given the necessary data, the algorithm then is the following:

1. For each planning increment in LIFTCAP,
    a. For each channel in planning increment that receives positive flow,

(1) Compute number of assets assigned to channel by the equation

$$N = \frac{R \times CT}{A}$$

where N = Number of assets on channel
R = Lift capability of channel (rate) as determined by LIFTCAP
CT = Cycle time of asset on channel
A = Asset capacity.

(2) For each MRMATE time period that falls within this planning increment, compute the "actual" cumulative flow occurring in the interval (T_0, T_P) where T_0 is the time at which the planning increment starts (days) and T_P is the time at which the current time period ends. For the initial delay, D, if in planning increment 1, use the transit time plus some arbitrary constant (may be zero or some positive number; used to assure that the first ships leaving POE will not do so before cargo may arrive at POE to load); if in other than first planning increment, set D = CT/(2xN), the average time needed for the first ship to be able to arrive at POD the first time. This computation will be performed according to the following equations:

$$Y(T_P) = INT((N/CT) \times (T_P - T_0 - D)) + 1$$

If this is zero, assign $F(T_P) = 0$ and go to next time period.

$$U(T_P) = INT((N/CT) \times (T_P - T_0 - D - OL)) + 1$$

$$TY(T_P) = D + \frac{(Y(T_P)-1) \times CT}{N}$$

$$F(T_P) = A \times \left(U(T_P) + \sum_{i=1}^{Y(T_P)-U(T_P)} \frac{(i-1) \times (CT/N) + T_P - TY(T_P)}{OL}\right)$$

If $Y(T_P)-U(T_P) = 0$, ignore the summation.

(3) For each MRMATE time period, compute actual flow within the time period according to

$$f(T_P) = F(T_P) - F(T_{P-1}).$$

If the time period is the first within the planning increment,

$f(T_p) = F(T_p)$.

2. For each POE and for each POD for each time period, check that port capability is not violated.

> THIS PORTION OF THE ALGORITHM WILL BE PROVIDED IN THE FUTURE. ITS OMISSION AT THIS TIME WILL NOT AFFECT THE FUNCTION OF THE ALGORITHM, HOWEVER, SINCE PORT CAPABILITIES ARE CURRENTLY SET TO INFINITY.

The above algorithm should be implemented as a subroutine to be called by the MRMATE driver. It can merely replace the subroutine that is currently used to assign lift capability to MRMATE nodes. An appendix is attached which describes the above algorithm in "pseudo-code" to facilitate its translation into Fortran.

4.0 ADVANTAGES AND DISADVANTAGES

The algorithm described above has several advantages and disadvantages. Among its advantages are the fact that it should add little to the time required to solve a deployment problem. It should be relatively easy to implement since the code should be straight-forward, and since it may merely replace an already-existing portion of the SCOPE code. It also will improve the accuracy of the reported closure estimate to the minimum of a) the length of an MRMATE time period and b) the maximum value of CT/N for all open channels.

Disadvantages include the fact that convergence of the model to the optimal solution may be lost. Also, the algorithm depends heavily on the assumption of the equal spacing of ships. Finally, by using long MRMATE time periods (say, one time period

of duration 180 days), long cycle times (say, 100 days), few assets (say, 1) and a small quantity of cargo to be moved (say, one ship-load), the model accuracy is still very poor. In general, though, reasonable accuracy for deployment problems will be provided.

APPENDIX - PSEUDO-CODE FOR MODIFICATIONS TO MMGDRV

Variable	Purpose
OL	ship offload time
A	average asset capacity
D	delay (transit time or CT/2N)
N	number of ships
TO	start day of given LIFTCAP time period
F	flow on LIFTCAP channel as determined by the LIFTCAP network optimizer.
Y(m)	number of ship arrivals between TO and MRMATE period m.
U(m)	number of ships offloaded between TO and MRMATE period m.
CT	channel cycle time
L	length in days from TO through MRMATE period m.
T(m)	day of last ship arrival in MRMATE period m.
FC(m)	cumulative flow (in tons) of channel capability from TO through MRMATE period m.
MRCAP(m)	capability of given channel in MRMATE period m.

```
OL = 2

FOR j = 1 TO number of sea LIFTCAP channels  BEGIN

     A = average asset capacity on channel j

     FOR k = 1 TO number of liftcap periods  BEGIN

          N = (F * CT) / A

          IF (J = 1)
               D = channel transit time
          ELSE
               D = CT / (2 * N)

          TO = start day of LIFTCAP period k

          FOR m = 1 TO number of MRMATE periods in LIFTCAP period
          k BEGIN
               L = (end day of MRMATE period m) - TO

               IF (L >= D)
                    Y(m) = (N/CT) * (L - D)
               ELSE
                    Y(m) = 0

               IF (L >= (D + OL) )
                    U(m) = (N/CT) * (L - D - OL)
               ELSE
                    U(m) = 0

               T(m) = D + ( (Y(m) - 1) * CT / N )

               FC(m) = A * U(m)

               IF (Y(m) > U(m) )  BEGIN
                    TEMP = 0
                    NET = Y(m) - U(m)
                    FOR i = 1 TO NET
                         TEMP = TEMP + ((i-1) * (CT/N)) + TO + L
                    FC(m) = FC(m) + TEMP
               END

          END

          MRCAP(1) = FC(1)

          FOR m = 2 TO number of MRMATE periods in LIFTCAP period
          k  BEGIN

               MRCAP(m) = FC(m) - FC(m-1)
          END
     END
END
```

E-24-6

Report for:
Joint Deployment Agency
MacDill Air Force Base, FL 33608

Disaggregation Issues in MRMATE

Ananth. V. Iyer
John J. Jarvis
William G. Nulty
H. Donald Ratliff
Michael A. Trick

PDRC 86-07

Report by:
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

This work is supported by the Office of Naval Research under Contract No. N0014-85-C-0797 and N00014-83-K-0147. Reproduction in whole or part is permitted for any purpose of the U.S. Government

DISAGGREGATION - AN APPROACH

This report discusses a heuristic procedure to disaggregate the MRMATE solution generated by SCOPE (MODES). MRMATE is solved with aggregate MRs (upto 1000 in a cluster) as source nodes and channels aggregated by planning period as sink nodes. Solution to MRMATE provides an allocation of these aggregate MRs to channels. Figure 1 shows an example MRMATE network along with its solution. There are two aggregate MRs in the example, aggregate MR 1 being an aggregate of detailed MRs a,b and c, and aggregate MR 2 being an aggregate of detailed MRs d,e and f. MRMATE solution allocates aggregate MRs to channels during each time period. In Figure 1, Ci,tj refers to the capability of channel i during time period j. The example has two time periods each being 5 units long. Channels 1 and 2 form route 1, while channels 3 and 4 form route 2.

Short Term Approach

The following outlines a short term procedure to disaggregate MRMATE solutions and provide an allocation of detailed MRs to POD - POE pairs (routes) along with an estimated closure date.

(1) For each route derive the total capability during each planning period available to an aggregate MR. This essentially involves adding MRMATE flows from this aggregate MR to channels between the same POE - POD pairs i.e. across assets between the same POE - POD pairs in a time period. In the example 5 + 10 = 15 units of capability (flows on arcs 1-c1,t1 and 1-c2,t1) is available for aggregate MR 1 on route 1 during time period 1 and 5 units of capability (flow on arc 1-c2,t2) is available for

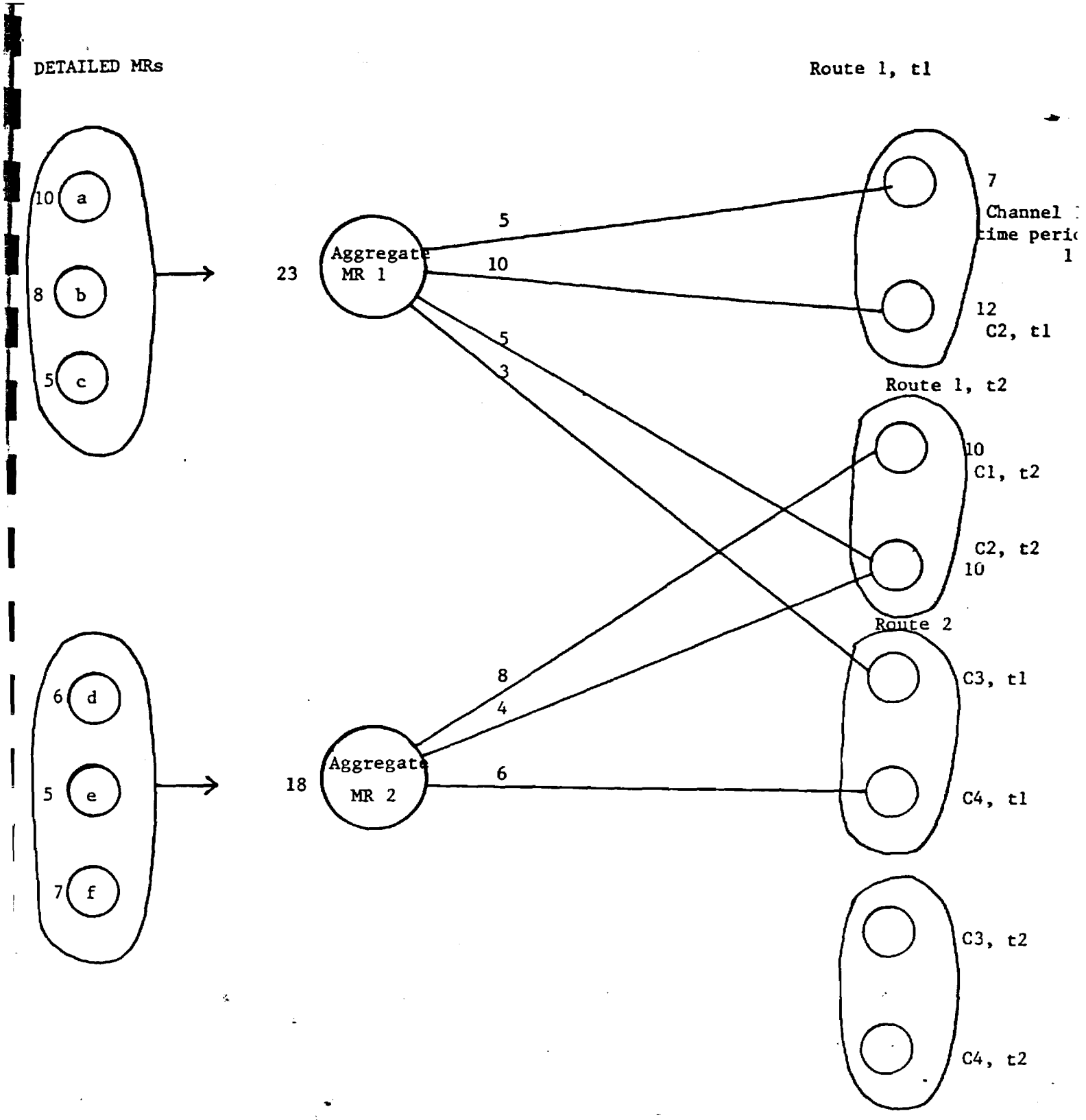


Figure 1 MRMATE information

aggregate MR 1 on route 2 in time period 2 as shown in Figure 2(a). Similarly the capability of aggregate MR - route pairs are derived for each time period as in Figures 2(b), 2(c) and 2(d).

(2) For each aggregate MR, examine the detailed MRs in the order of their mean time window value (i.e.(EAD + LAD)/ 2). The estimated closure of a MR allocated to a feasible route is given as

trj + (MR i / route r,j) where MR i belongs to aggregate MR j and trj is the current closure date for aggregate MR j on route r

route (r,j) = sum of flows from MR j to route r channels

A strategy might be to examine all feasible routes and allocate an MR to the route that permits earliest closure. This requires that all feasible routes be examined for an aggregate MR (that the detailed MR belongs to) to pick out that route with earliest closure. An alternative strategy would be to allocate the MR to the first route that permits closure within the planning horizion.

Consider MR a which belongs to aggregate MR 1 in the example in Figure 1.

Closure of MR a on route 1 = MR quantity / (route1 - aggregate MR 1 capability)

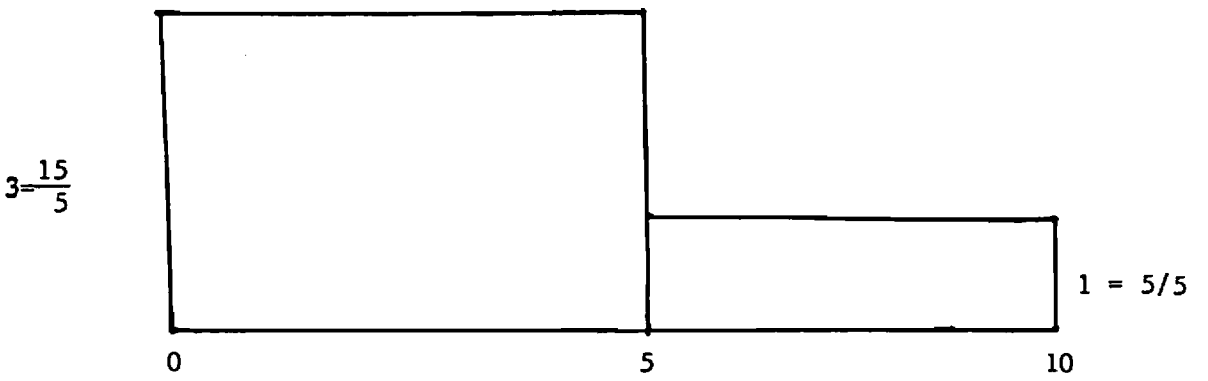
= 10 / 3

= 3.33

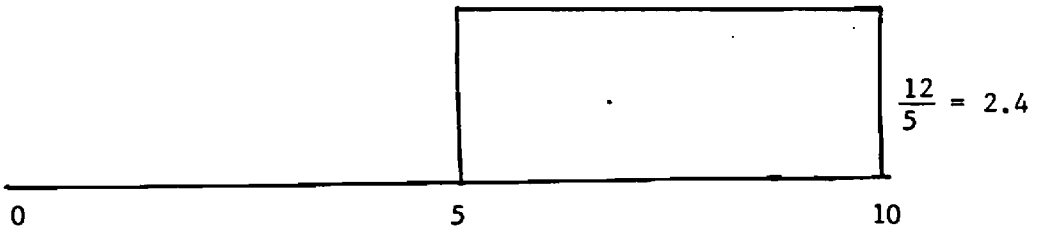
Closure of MR a on route 2 = 5 + 10 / 2.4

= 9.16

Using the earliest closure route selection strategy route 1 is selected for MR a. This sets t11 = 3.33. Consider MR b which

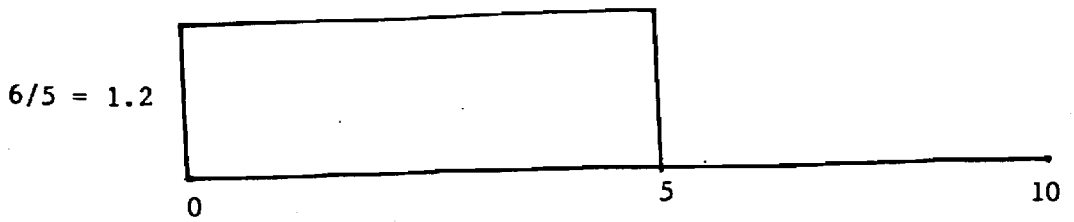


ROUTE 1 AGGREGATE MR 1



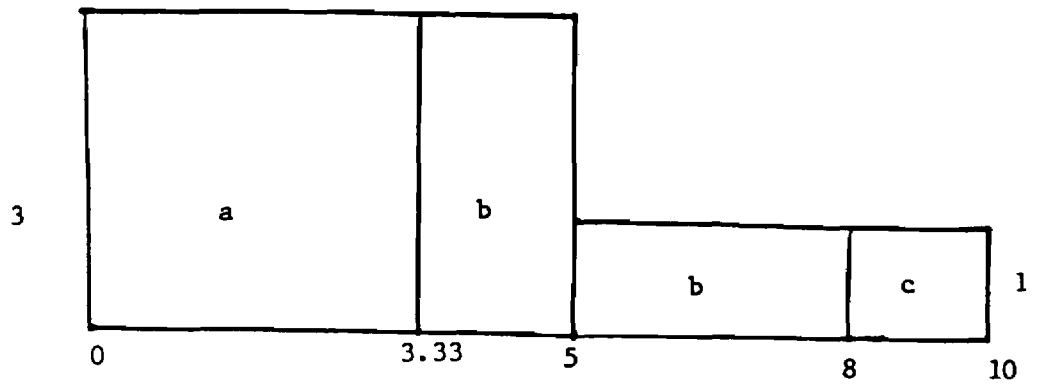
ROUTE 1 AGGREGATE MR 2

ROUTE 2 AGGREGATE MR 1

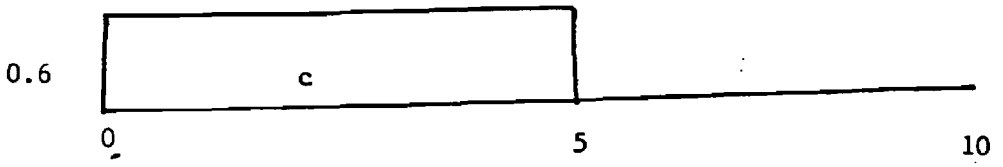


ROUTE 2 AGGREGATE MR 2

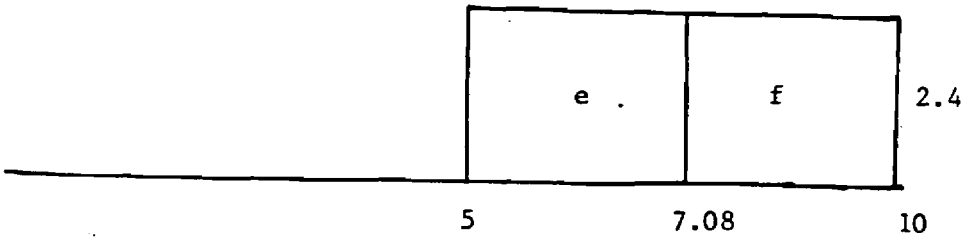
FIGURE 2 Route Capability



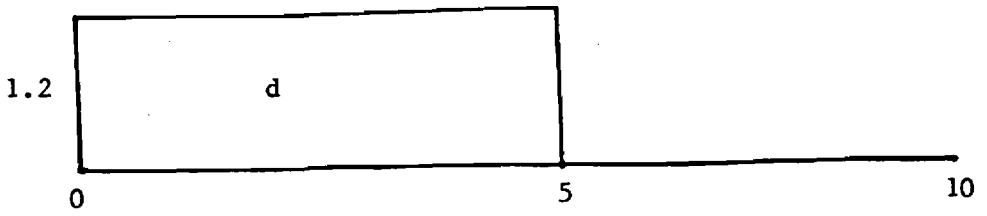
3(a) ROUTE 1 AGGREGATE MR 1 ALLOCATION



3(b) ROUTE 2 AGGREGATE MR 1 ALLOCATION



3(c) ROUTE 1 AGGREGATE MR 2 ALLOCATION



3(d) ROUTE 2 AGGREGATE MR 2 ALLOCATION

Figure 3

MR	Route Allocated	Closure Date
a	1	3.33
b	1	8
c	1,2	10
d	2	5
e	1	7.08
f	1	10

Figure 4 MR route allocation and closure dates in step 2

MR	Loading Start Date	Closure Date	
a	1	4	
b	4	9	
c	1	11	(* Split across routes *)
d	1	5	
e	5	8	
f	8	11	

Figure 5 MR closure dates on routes

(4) The last step that can be performed is to increase the capability available on a route for an MR. Its effect is to permit earlier closure of MRs. This involves allocating the slack lift capability available at routes to aggregate MRs that utilize this route.

One strategy for slack capability allocation is to allocate it proportional to the flows from the aggregate MR to the route. However, this procedure can be carried out in step 1, by adjusting the channel capability available to an aggregate MR by the proportional amount of slack capability. In the example it means that

Aggregate MR 1 - C1,t1 capability = 7 units

Aggregate MR 1 - C2,t2 capability = 5 + (5 / 9) * (10 - 9)

= 5.55 units

Advantages of this approach

One desirable feature of this heuristic is that it allocates a MR contigously across time. This implies that once loading of an MR begins it continues till the MR is completed, within each time period. For MRs spread across time periods, there may be an idle time interrupting MR loading due to no capability being available in an intermediate time period.

Furthermore, the computational time is distinctly reduced as discussed in the next section.

Long Term alternatives

A desirable procedure for disaggregation of the MRMATE solution would build an optimization model for the entire process and attempt to generate solution techniques that would gaurantee

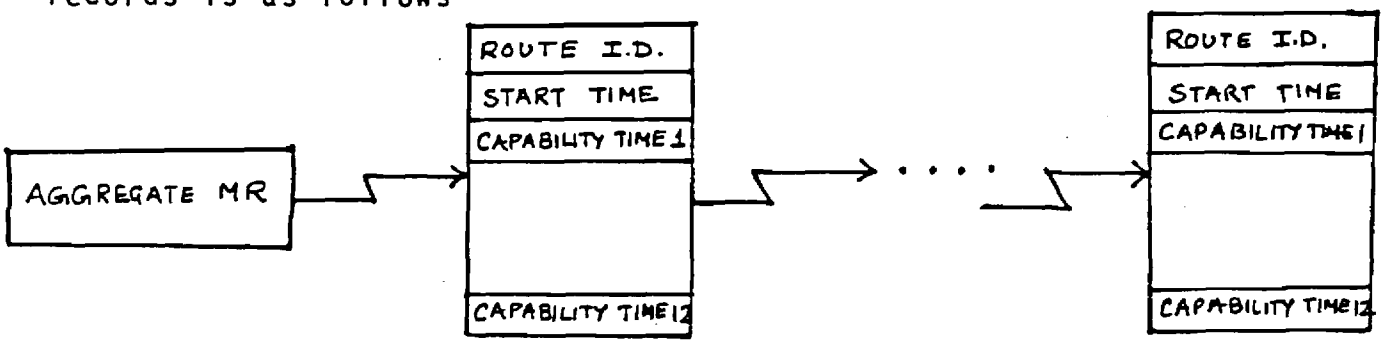
convergence to a solution of high quality. One approach is to use a three phase approach to solve the MRMATE problem where an aggregate MR to channel allocation would be disaggregated by another network flow model to generate information thet would be used to re-solve the aggregate model. Such an approach is discussed in PDRC Report 85-09. However, the three phase approach does not include the constraint that each MR should be allocated only one route. Procedures need to be devised to include this constraint in the model and ensuing solution procedures.

COMPUTATIONAL DETAILS

Input to this procedure is the solution to the MRMATE problem. We assume that for each aggregate MR there are atmost five feasible routes. MRMATE is assumed to have 100 aggregate MRs with each aggregate MR composed of less than 1000 detailed MRs.

SPACE REQUIREMENTS

We set up the data structure as follows. For each aggregate MR we have a linked list. Each component of the linked list is a feasible route for the aggregate MR. For each time period, the capability of this route allocated to an aggregate MR is obtained by adding flows from the aggregate MR to channels,in the time period, that are between the POE - POD pair of this route. Thus a sample linked list for an aggregate MR and the associated records is as follows



Thus the amount of information required is

Aggregate MRs x Feasible routes per MR x Time periods

= 100 x 5 x 12 = 6000

Correspondingly for the CSC disaggregation there are

Aggregate MRs x Feasible routes x Planning horizon

= 100 x 5 x 180 = 90000 to be stored. Assuming 4 bytes per unit of information to be stored in core this implies that

GT heuristic space required = 6000 x 4 = 24000 bytes (24 K)

CSC heuristic space required = 90000 x 4 = 360000 bytes (360 K)

Therefore it is reasonable to expect that we could store all required information in GT heuristic in core as against the CSC heuristic. This implies that I/O time spent in retrieving information from disk would be saved.

TIME REQUIRED

The GT heuristic procedure would work as follows. For each detailed MR, we know the corresponding aggregate MR it belongs to. We start with the first feasible route on the linked list for the aggregate MR. Traversing down the time periods for the route, we calculate expected closure date. This is done as follows

For the route under consideration

Read the current start time

Go down to the time period that contains this start time

Calculate closure with this capability as

Closure Time = Start time + (MR size / capability during this time period)

If closure is less than the end point of this time period, set closure for this MR on this route = closure time.

Else fill up this time period and subtract the amount of the MR already allocated i.e. set the MR quantity = MR - (end of time period - start time) x capability available. Set start date = End of this time period. Repeat.

Thus for each detailed MR, for each route we have to compute the closure during a time period for atmost 12 time periods. Updating the information corresponds to setting the start time for this route as the closure time for the last MR assigned to

this route. Furthermore, if we were to choose the route with the earliest closure for a detailed MR, then atmost five route would have to be examined, the route information locations identified by the linked lists associated with each aggregate MR. This would imply that we calculate closure, for GT heuristic, atmost 12 x 5 = 60 times for each detailed MR.

CSC heuristic requires closure to be calculated by examining the route capability available per day for the planning horizon (180 days). This implies
180 x 5 = 900 calculations for each detailed MR.

This implies a time savings factor of 900 / 60 = 15

Also we do not disaggregate a channel across time to determine the channel capability by day allocated to an aggregate MR, and hence do not calculate the available capability by day for each route. Deleting this procedure would also save time considerably.

If (as reported) CSC disaggregation would have the information stored on disk, while the GT heuristic enables all information to be stored in core, we expect based on the relative amounts of information stored a saving in time by a factor of (90000 / 6000) = 15.

Thus if we consider the combined potential time savings factor
= 15 x 15 = 225.

Code and Logic Modifications

Current Code Structure

Briefly, the heuristic is composed of seven steps. The first five perform preprocessing, and the last two schedule unaggregated movement requirements. It was established from the 4102 run that the heuristic was scheduling movement requirements at the rate of 425 movement requirements per hour. At this rate the heuristic would take seven days to schedule the 4102 plan. Steps 6 and 7 were clearly unacceptable in running time, and were the focus of the Georgia Tech effort.

The current logic of steps 6 and 7 are:

```
For each unaggregated movement requirement
        Bring all feasible routes for the corresponding
        aggregated movement requirement from disk into memory.

        Select the best route.
```

Computational Requirements

For the 4102 problem, assuming five feasible routes for each movement requirement, a total of (68000 x 5 x 13000 bytes/route =) 4.4 billion disk byte operations are performed for the current logic.

Code Modification Suggestions

We recommend the following logic:

> Sort the unaggregated movement requirements first by aggregate ID, then by priority, and finally by time window midpoint or RDD. (This step should be done in the preprocessor).
>
> For each aggregated movement requirement
>
> > Bring all feasible routes for the current aggregated movement requirement from disk into memory.
> >
> > Sequentially process the sorted unaggregated movement requirements corresponding to the aggregated movement requirement (select the best route).

Computational Requirements for Modified Logic

For the 4102 plan, this logic would perform (60 aggregated movement requirements x 5 x 13000 =) 3.9 million byte operations, a thousand-fold reduction from the current logic.

We believe this change in logic will result in a major increase in heuristic speed, and should reduce the seven day run time down to 1-2 hours.

Interface with ship capability adjustment

A companion PDRC report discusses a preprocessor to MRMATE that adjusts channel capabilities in MRMATE based on decisions regarding ship scheduling within each channel. The motivation for this adjustment was to reflect channel utilization more realistically across time. However, since MRMATE channels are aggregated across time periods, the information generated is at the level of detail of a time period in MRMATE.

If a heuristic such as that described in Section 1 is used, then the effect is to allocate uniform capability across a time period for a channel. Thus, if a large MRMATE time period were present, the same difficulty as before is observed i.e. closure is spread out across time even though the actual ship arrives at point in time delivers materiel at some time which is smaller than the time period.

Short Term approach

This section discusses some short term approaches to resolve the problem. One approach is to consider the number of ships and their capacities in each channel that compose route capability in a time period. This information is used to derive an average ship capacity as follows

n_i = number of ships in channel i

ci = capacity of the ship in channel i

Average ship capacity on the route =

sum (ni * ci)/ sum ni

Number of ships on the route = sum ni

The number of ships in a route are considered to each have

the average ship capacity. If it is assumed that the ships in a route are uniformly spaced out within a time period, an adjustment similar to that for channel capability can be carried out to determine the route capability by day of an MRMATE time period. However, this information need not be used explicitly to derive a route capability per day. Instead the heuristic is implemented as in Section 1 and closure dates are adjusted to reflect the spikes in capability due to ship arrivals.

This procedure is an approximation because it ignores relations across time periods as individual ships and their capacities are not tracked across time. However, it does maintain route capability across a time period as the number of ships in a route times the average ship capacity is exactly the route capability during the time period under consideration. Also the procedure sometimes overestimates and sometimes underestimates the actual closure dates because actual ship capacities are ignored.

Long Term approaches

The main difficulty with the short term procedure is that it does not track individual ships. This reveals a crucial decision to the disaggregation procedure namely that of scheduling individual ships across the planning horizon. If individual ships were allocated schedules, then the spikes generated and the disaggregation decisions affected would be co-ordinated across time periods so that the closure dates generated would be good estimates of expected closure dates. However the ships schedule would in turn depend on the MRs loaded on it which is the

decision to be made by the model. Therefore a reasonable long term approach would be an iterative procedure, iterating between setting ship schedules and allocating MRs to routes, and using information generated at each iteration to improve ship schedules and MR route allocation in order to minimize the total penalty incurred due to MRs closing outside their time windows.